UNIVERSIDAD AUTONOMA DE MADRID

Escuela Politécnica Superior

19|20

Trabajo fin de grado

Estudio de secuencias en un dominio de películas y su aplicación en sistemas de recomendación

Adrián Iglesias Ortiz

Escuela Politécnica Superior
Universidad Autónoma de Madrid
C\Francisco Tomás y Valiente nº 11

Campus Internacional excelencia UAM CSIC+

www.uam.es

UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Estudio de secuencias en un dominio de películas y su aplicación en sistemas de recomendación

Autor: Adrián Iglesias Ortiz
Tutor: Alejandro Bellogín Kouki
Ponente: Iván Cantador Gutirrez

septiembre 2019

Adrián Iglesias Ortiz
Estudio de secuencias en un dominio de películas y su aplicación en sistemas de recomendación

Adrián Iglesias Ortiz
C\ Francisco Tomás y Valiente Nº 11

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

Todas las verdades son fáciles de entender una vez descubiertas,
el punto es descubrirlas.

Galileo Galilei

Agradecimientos

En primer lugar quiero agradecer a toda mi familia y a mi pareja Irene todo el apoyo que me han dado en esta última etapa de la carrera. Su apoyo ha sido indispensable, sobretodo en los momentos mas difíciles y siempre han respetado mis decisiones y me han ayudado.

También quiero dar las gracias a mi tutor Alejandro Bellogín por toda la ayuda prestada en la elaboración de este proyecto. Su respaldo y consejo han sido de suma importancia para poder llevar a cabo con éxito este trabajo. Aunque no he podido coincidir con el hasta la elaboración de este proyecto, ha sido uno de los mejores profesores que he tenido en toda la carrera.

Por último, dar también las gracias a mis amigos y compañeros de la univerdad, con los que he compartido muchas horas tanto dentro como fuera de la facultad.

Resumen

En la actualidad, los sistemas de recomendación son utilizados en una inmensa cantidad de aplicaciones. Esto se debe a que permiten ofrecer a las personas productos, películas, viajes, etc., basándose en las necesidades potenciales de estas. Estas necesidades se obtienen de diferentes maneras; en base a sus propias búsquedas, en base a las búsquedas de otras personas con las que estas guardan alguna relación, más populares, etc. En definitiva permiten a las aplicaciones poder acertar a la hora de recomendar sus productos empleando la propia información que ofrecen los usuarios.

Por ello este Trabajo de Fin de Grado se centra en el estudio de una de estas mejoras para un sistema de recomendación de películas. De este modo analizaremos el patrón que siguen los usuarios a la hora de visualizar las películas que pertenecen a una saga. Extraeremos datos de la web mediante técnicas de crawling sobre los usuarios de una página web de puntuación de películas y se analizará la forma en la que los usuarios han visualizado el orden de dichas películas basándose en sus valoraciones y fechas en las que las valoraron.

Para este trabajo nos hemos centrado en la página web de IMDb, la cual es la que mayor número de datos de usuarios, valoraciones y películas dispone. De modo que realizando una extracción y recopilación de datos de películas y de las películas asociadas a estas (pertenecientes a la misma saga, precuelas, remakes...), y de los usuarios y sus valoraciones sobre estas películas, podemos estudiar los patrones de visualización mediante una serie de algoritmos.

El algoritmo principal y en el que se basa este Trabajo de Fin de Grado es el algoritmo Smith-Waterman. Este algoritmo es una reconocida estrategia para realizar alineamiento local de secuencias; es decir que encuentra regiones similares entre un par de secuencias. Es un algoritmo utilizado para encontrar secuencias biológicas como ADN, ARN o proteínas, pero que empleado de manera adecuada puede encontrar patrones entre las películas que ha visto un usuario y la secuencia de películas de una saga, para comprobar así con la frecuencia con la que los usuarios ven las películas de una saga, relativamente larga, en orden y seguidas. De esta manera se pueden estudiar cuántos usuarios ven las películas de una saga de 3, 4 o más películas en un orden u otro, ayudando así a mejorar la recomendación de películas a los usuarios basándose en los patrones de visualización más comunes.

Palabras clave

Sistema de recomendación, algoritmo , secuencias, subsecuencias, crawling, aplicación

Abstract

Currently, recommendation systems are used in a huge number of applications. This is because they allow people to offer products, movies, trips, etc., based on their potential needs. These needs are obtained in different ways; based on their own searches, based on the searches of other people with whom they have a relationship, more popular, etc. In short, they allow applications to be successful when recommending their products using the information offered by users.

Therefore, this Final Degree Project focuses on the study of one of these improvements for a movie recommendation system. In this way we will analyze the pattern that users follow when viewing movies that belong to a saga. We will extract data from the web using crawling techniques on the users of a movie scoring web page and we will analyze the way in which users have visualized the order of said movies based on their ratings and dates on which they rated them.

For this work we have focused on the IMDb website, which has the highest number of user data, ratings and movies available. So, by extracting and collecting data from movies and movies associated with them (belonging to the same saga, prequels, remakes ...), and users and their ratings on these films, we can study the display patterns by using a series of algorithms.

The main algorithm – and which this Final Degree Project is based on – is the Smith-Waterman algorithm. This algorithm is a recognized strategy to perform local sequence alignment; that is to say that it finds similar regions between a pair of sequences. It is an algorithm used to find biological sequences such as DNA, RNA or proteins, but that properly used can find patterns between the films that a user has seen and the sequence of films in a saga, to check the frequency with which users watch movies in a series, relatively long, in order and in a row. This way we can study how many users watch movies from a series of 3, 4 or more movies in one order or another, thus helping to improve the recommendation of movies to users based on the most common viewing patterns.

Keywords

Recommender system, algorithm, sequences, subsequences, crawling, application

Índice

Listas

Lista de figuras

Lista de tablas

1

Introducción

La recomendación automática en la red es un elemento de suma importancia actualmente. Desde que el uso de Internet comenzó a extenderse en 1990 por todo el mundo, se estima que en la actualidad hay 4.536 millones de usuarios con conexión a la red [1]. Las empresas son conscientes de la importancia que tiene el mercado online, hasta el punto de que algunas empresas únicamente disponen de sitio web y no poseen sede física.

Debido a la inmensa cantidad de usuarios que emplean la red para consumir diferentes tipos de productos, la necesidad de las empresas por recomendar de manera acertada sus productos a los usuarios se ha ido incrementando con los años. Es aquí donde entran en juego los sistemas de recomendación.

El enorme potencial de estos Sistemas de recomendación (SR) es por lo que actualmente muchas empresas tratan de mejorarlos, ya que si consigues recomendar productos de manera acertada y facilitas el acceso a estos a los usuarios de tu aplicación, es lógico pensar que se incrementará el número de posibles ventas.

Un claro y notorio ejemplo de esto es el concurso lanzado por Netflix en 2006 en el que se pretendía mejorar el acierto del algoritmo de recomendación automática para su tienda de alquiler de películas, en un 10 %. El objetivo era mejorar o reescribir un algoritmo que superase al que empleaban en aquel momento, llamado Cinematch. Para ello disponían de un histórico de datos basado en las puntuaciones que los usuarios daban a las películas [2]. El premio era de 1 millón de dolares para el equipo que lo lograse. Siendo conscientes de que, por aquellas fechas, a Netflix su plataforma de alquiler de películas le facturaba a la empresa una media de 100 millones de dolares anuales, incrementar en un 10 % las ventas supondría aumentar en 10 millones sus ganancias. Por lo tanto, un premio de 1 millón de dolares es algo más que razonable.

En este trabajo pretendemos hacer un estudio similar de un histórico de datos de usuarios y películas. Pero en lugar de basarnos en la puntuación, nos centraremos en el orden en el que los usuarios ven las películas. Pudiendo así predecir qué películas se deberían recomendar a los usuarios dependiendo de la longitud de la saga a la que pertenece la película y la manera promedio que tienen los usuarios de ver sagas de tamaño similar. Hemos de decir que, en este proyecto, no se realizarán re-

comendaciones concretas a usuarios, únicamente se estudiarán los datos y la forma en la que los usuarios visualizan las películas.

1.1. Motivación

Como ya comentamos antes, a medida que un mayor número de usuarios tiene conexión a la red, las necesidades de las empresas de automatizar de manera eficiente la recomendación de sus productos son mucho mayores. Es por esto que el estudio de estas herramientas y de los datos que emplean tiene una mayor demanda en el mundo actual. La parte de recopilación de datos y el análisis de estos es un sector en auge actualmente.

En este trabajo se pretende cubrir todo este proceso, desde la recopilación de datos mediante técnicas de crawling, al tratamiento de los datos, para su procesamiento posterior mediante el uso de algoritmos de estudio de secuencias.

Debido a que el estudio de sistemas de recomendación es algo que está muy extendido actualmente, queríamos centrarnos en algo más concreto de este sector. Específicamente, en el análisis de patrones en los datos para poder responder preguntas como ¿cuántos usuarios han visto la 1º, 2º y 3º película de una saga?. Analizando así posteriormente las razones de porqué no se ha visto la 4º cuando sí se han visto las anteriores o, en general, analizar si los usuarios tienden a ver las películas en el mismo orden en el que una saga está definida o no. De este modo se pretende observar los datos para responder preguntas que permitan mejorar, no solo la recomendación de películas, si no también el modo en el que se lanzan las películas de estreno a las plataformas o la publicidad que se invierte en estas.

1.2. Objetivos

El objetivo principal de este trabajo es, por tanto, la recopilación de datos y el uso de nuevas técnicas basadas en el estudio de secuencias con el algoritmo Smith-Waterman, para poder analizar y comprender los patrones de visualización de películas de los usuarios de una plataforma web.

Se realizará un estudio de la aplicación del algoritmo Smith-Waterman sobre una serie de datasets con distintos pesos y variables para observar las diferencias a la hora de detectar subsecuencias comunes entre las películas vistas por el usuario y una determinada secuencia **canónica** de películas. Estas secuencias canónicas son las películas pertenecientes a una saga, en el orden definido por, en este caso, IMDb. Generalmente el orden canónico de una saga suele ser el orden cronológico, aunque en algunos casos esto no es así.

Del mismo modo, se generarán resultados cuya distribución se ajuste a un modelo probabilístico,

los cuales puedan ser utilizados en futuros contrastes de hipótesis. Permitiendo, de esta manera, responder a cuestiones que ayuden a mejorar los sistemas de recomendación de películas pertenecientes a una misma saga o franquicia.

1.3. Organización de la memoria

Se ha estructurado este documento en cinco capítulos principales, los cuales son los siguientes:

Introducción: Capítulo que sirve de preludio y en el que ya hemos visto el por qué este proyecto tiene una importante utilidad actual y qué necesidades pretende estudiar y resolver.

Estado del arte: Se estudiará la información que hay actualmente sobre los SR, el algoritmo Smith-Waterman y cómo este puede mejorar la recomendación de determinados productos. Del mismo modo se comentarán las técnicas de crawling y su utilidad en los SR.

Diseño y Desarrollo: Basándonos en todas las herramientas comentadas en la sección de estado del arte, indicaremos detalladamente cómo desarrollar dichas técnicas. Del mismo modo, indicaremos todos los pasos seguidos para la implementación del algoritmo y la elaboración del dataset y las tecnologías empleadas para su desarrollo.

Pruebas y resultados: Una vez terminada la implementación del algoritmo y aplicado sobre los diferentes datasets, se realizarán distintas pruebas con distintos pesos y valores, detallando y analizando los resultados obtenidos.

Conclusiones y trabajo futuro: En base a las pruebas, se plantearán posibles mejoras o posibles aplicaciones futuras de dicho algoritmo sobre otros conjuntos de datos y las mejoras que supondrían a otros SR.

2

Estado del arte

En esta sección veremos la situación actual de las diferentes areas tecnológicas que son relevantes para la realización de este proyecto.

En este trabajo emplearemos herramientas tanto para la recopilación de datos desde la web como para su uso y estudio, con el objetivo de analizar patrones en el comportamiento de los usuarios en la forma de visualizar las película de una saga. Todas las herramientas empleadas en este proyecto y sus alternativas actuales, serán vistas en este capítulo.

2.1. Sistemas de recomendación (SR)

Hoy en día, nos enfrentamos a un sinfín de opciones a la hora de elegir cualquier producto. También es una situación muy común que, debido al abrumador número de opciones, no logremos decidir con que quedarnos y al final no escojamos ninguna de dichas opciones. Este hecho sucede con más frecuencia en algunos productos que en otros y en el caso de las series y películas sucede habitualmente ya qué, en la actualidad, comenzar a ver una serie o una película implica invertir un tiempo que, a día de hoy, es un bien escaso.

De este modo, es importante ser capaces de recomendar al usuario productos de una manera relevante y que mejoren su experiencia. Es decir, hacerle la vida mas sencilla a los usuarios. Es aquí donde entran en juego los sistemas de recomendación.

2.1.1. ¿Qué es un sistema de recomendación?

Un sistema de recomendación es en definitiva una serie de algoritmos que finalmente muestran información relevante a nuestros intereses [3]. Para realizar dichas recomendaciones, el sistema analiza datos e información histórica de los usuarios (edad, puntuaciones, compras realizadas previamente, etc) y de los ítems del sistema (precios, contenidos similares, distintas marcas, etc). Por último, utilizando dicha información, predice qué producto puede resultar interesante para el usuario.

En la actualidad, muchos de los sistemas que usamos utilizan este principio. Según un estudio de Mckinsey & Company en el 2013, el 35 % de los ingresos del sitio web de Amazon, corresponden a recomendaciones de productos generadas por su sistema de recomendación.

En definitiva el objetivo de un SR es predecir o estimar la importancia de un determinado ítem para un usuario concreto. Esto lo logra el SR mediante el uso de una serie de algoritmos que minimizan la preferencia esperada contra la preferencia real que tiene el usuario por dicho ítem. Daremos una breve explicación de los principales métodos que, actualmente, son empleados por los SR para la recomendación de ítems a los usuarios.

SR basados en contenido

Estos SR se basan en la regla principal de la intuición, por lo que el usuario recibirá información similar a la que ha mostrado interés previamente, independientemente de las opiniones del resto de usuarios. Es decir se recomienda a cada usuario de manera individual, sin mirar a los demás, de modo que utilizan el producto y no el usuario como base de la predicción. De esta manera los ítems tienen un espacio de características (autor, lugar, idioma. . .) a las cuales se les asigna un determinado valor. Con este espacio de datos, se utiliza un algoritmo muy empleado en el sector de la recomendación que es el basado en vecinos próximos o K-neares neighbours (kNN) . Este algoritmo se utiliza frecuentemente para realizar recomendaciones basadas en similitudes, en este caso entre los ítems. Estas similitudes se pueden calcular empleando una función de similitud (por ejemplo coseno).

SR de filtrado colaborativo

En los SR de filtrado colaborativo, los usuarios se benefician de la experiencia de otros usuarios. Este sistema analiza las compras, preferencias, importe de las compras, calificaciones que el usuario ha dado de los distintos productos, etc y busca otros usuarios parecidos a él y que han tomado decisiones similares. De este modo los productos que han tenido éxito para dichos usuarios posiblemente también le interesarán al nuevo usuario. Del mismo modo que en los SR basados en contenido, uno de los métodos mas utilizados es el de *kNN* basado en similitud entre usuarios o ítems. La diferencia reside en que la similitud entre ítems se basa siempre en ratings y que en las recomendaciones a usuario intervienen ratings de otros usuarios. El filtrado colaborativo tiene dos perspectivas (cuasi)simétricas que son recomendar al usuario los ítems que le han gustado a usuarios similares o recomendar al usuario ítems que se parecen a otros ítems que le han gustado a este. Por ello se basa en dos premisas fundamentales: basado en usuarios y basado en ítems.

2.2. Búsqueda y recopilación de datos en la web

Como ya sabemos, en la actualidad, la web contiene una gran cantidad de información personal sobre sus usuarios y mucha de esa información es accesible para todo el mundo y refleja gran parte de los gustos de los usuarios en un determinado sector. Todas las calificaciones, puntuaciones, comentarios y opiniones de los usuarios en sitios web públicos son accesibles para los usuarios de estos sitios web y en el mundo actual en el que la información es un bien muy valioso, pueden resultar de suma utilidad para algunas empresas.

Es por eso que una parte de este proyecto queríamos que fuese dedicada a la recopilación de dicha información y las herramientas que se han empleado para obtenerla.

2.2.1. Crawling

Los **Web crawler**, en ocasiones también llamados araña web o rastreador web, son unos programas que buscan de manera sistemática dentro de la web, desplazándose entre las distintas páginas web. El objetivo principal de este tipo de programas suele ser el de indexar páginas web para los diferentes navegadores o motores de búsqueda para actualizar así el contenido web de estos o indexar nuevos sitios web [4].

El proceso es simple, el web crawler empieza descargando y analizando una serie de páginas web de una lista de *URLs*, en inglés Uniform Resource Locator, proporcionada (llamado comúnmente *semillas*), identifica los hiperenlaces que hay en estas y los añade a la lista de *URL* a visitar. Luego descarga el contenido de estas nuevas páginas, analiza sus enlaces y así de manera recursiva.

El procedimiento es simple, pero conseguir un web crawler eficaz, que permita indexar un gran número de páginas diferentes y evite caer en bucles es complejo. Por ello se estipulan una serie de reglas en la forma en la que el crawler añade y explora las diferentes *URL* de su lista. Comúnmente se asigna un determinado nivel de profundidad para evitar que indexe demasiadas páginas del mismo tipo que no aporten nueva información. También se le suele dar prioridad a las *URL* que no contienen caracteres como *&,=,#,etc...* ya que suelen ser consultas dentro de la página principal o algún elemento dentro de la misma página.

Como ya dijimos antes entre las tareas más comunes de estos web crawlers son las de crear el índice para el motor de búsqueda, analizar los enlaces de un sitio para buscar qué enlaces ya no existen (rotos) y recopilar información de un cierto tipo, como precios o puntuaciones.

Esta última tarea es por la que estos web crawlers resultan útiles e interesantes para nuestro proyecto, ya que nos permiten recopilar puntuaciones y ratings de distintos usuarios en un mismo sitio web. Por esta razón los web crawlers son herramientas comúnmente empleadas junto con los SR.

2.3. Problema de subsecuencia común más larga

El problema de la subsecuencia común más larga o más conocido en inglés como Longest common subsequence (LCS) , es el problema de encontrar la subsecuencia más larga que es común en un conjunto de secuencias. A diferencia de las subcadenas (en ingles: substrings), las subsecuencias no necesitan tener posiciones consecutivas en la secuencia principal. El problema *LCS* es la base de los programas empleados para la comparación de datos y en los sistemas de control de versiones, como Git.

Si el número de secuencias de entrada es aleatorio, el problema puede llegar a ser *NP-completo*. Pero si el número de secuencias es constante, el problema puede resolverse en tiempo polinómico dividiendo el problema en subproblemas más pequeños y sencillos hasta que la solución sea trivial. De este modo cuando se quieren comparar unicamente dos secuencias, el problema *LCS* dividido en subproblemas más simples, es óptimo.

La función *LCS* para dos secuencias se define de la siguiente manera:

$$LCS(X_i, Y_j) = \begin{cases} \emptyset & \text{if } i = 0 \text{ or } j = 0 \\ LCS(X_{i-1}, Y_{j-1}) \vee x_i & \text{if } x_i = y_j \\ \text{longest}(LCS(X_i, Y_{j-1}), LCS(X_{i-1}, Y_j)) & \text{if } x_i \neq y_j \end{cases}$$

Donde X = (x_1,x_2...x_n) e Y = (y_1,y_2... y_n) son las dos secuencias a comparar y *LCS(X_i,Y_j)* representa la subsecuencia común mas larga de las dos secuencias X e Y. $LCS(X_{i-1}, Y_{j-1}) \vee x_i$ indica que se añade el valor de la cadena X en la posición i a la cadena *LCS*.

Esta función indica que para encontrar esta subsecuencia común más larga, comparamos los elementos x_i e y_j. Si estos son iguales, entonces se le añade a la subsecuencia *LCS(X_{i-1},Y_{j-1})* el elemento x_i, como se indica en la segunda propiedad de la función (inicialmente la subsecuencia *LCS(X_{i-1},Y_{j-1})* esta formada únicamente por $\emptyset$). En caso de que los dos elementos sean diferentes, entonces el resultado es la cadena mas larga entre *LCS(X_i,Y_{j-1})* y *LCS(X_{i-1},Y_j)*, como indica la tercera propiedad de la función.

Veamos el funcionamiento de este algoritmo obteniendo la subsecuencia común mas larga de dos secuencias sencillas, siendo X = (GAC) e Y = (AGCAT) las secuencias a comparar.

De este modo la tabla inicial quedaría de la siguiente manera:

La primera linea y la primera columna de la tabla tienen el valor $\emptyset$ de manera inicial, debido a que cuando un valor 0 se compara con otro valor (0 con AGCAT), el *LCS* siempre será $\emptyset$ por la primera propiedad de la función.

A continuación determinamos el *LCS(X_1,Y_1)*, comparando el primer elemento de cada secuencia,

	0	A	G	C	A	T
0	∅	∅	∅	∅	∅	∅
G	∅					
A	∅					
C	∅					

Tabla 2.1: Tabla de ejemplo *LCS*

es decir G y A. Al no ser iguales el *LCS* se calcula con la tercera propiedad de la función es decir con el *LCS* de mayor longitud de $LCS(X_1,Y_0)$ y $LCS(X_0,Y_1)$. Al ser ambos ∅, el valor de $LCS(X_1,Y_1)$ es ∅.

Calculamos ahora el $LCS(X_1,Y_2)$ comparando G con G. Estos valores sí son iguales por lo que empleamos la segunda propiedad de la función, agregando el valor G, es decir x_1, a la secuencia $LCS(X_0,Y_1)$ (la celda que está ubicada arriba a la izquierda de la posición actual que estamos comparando) la cual es (∅) y pasa a ser (G). Se calcula a continuación el $LCS(X_1,Y_3)$ comparando G con C, como son distintos el valor de $LCS(X_1,Y_3)$ es el *LCS* de mayor longitud de $LCS(X_1,Y_2)$ y $LCS(X_0,Y_3)$. Como el valor de arriba es ∅ y el valor de la izquierda es G, la secuencia más larga sería $LCS(X_0,Y_3)$ con lo que el valor para $LCS(X_1,Y_3)$ es G. Repetimos el proceso con el resto de la fila y la tabla quedaría de la siguiente forma (las flechas de la tabla indican la celda que de la que toma el valor la celda actual).

	0	A	G	C	A	T
0	∅	∅	∅	∅	∅	∅
G	∅	$\leftarrow^{\uparrow}$ ∅	$\nwarrow$ (G)	$\leftarrow$ (G)	$\leftarrow$ (G)	$\leftarrow$ (G)
A	∅					
C	∅					

Tabla 2.2: Fila G completada

Seguimos el mismo procedimiento fila por fila hasta completar la tabla. Cuando calculemos un $LCS(X_i,Y_j)$ en el que las comparaciones de sus elementos sean diferentes y el valor *LCS* de la celda a la izquierda y el de la celda superior sean de la misma longitud pero con valores distintos, el valor de $LCS(X_i,Y_j)$ son los dos distintos valores. Por ejemplo para $LCS(X_2,Y_2)$, el valor *LCS* sería (A) y (G). Del mismo modo si una celda tiene como valor *LCS* más de una subsequencia, cualquier nuevo valor que se le anexe a dicha celda se le añade a todas sus *LCS*.

De este modo repitiendo el procedimiento para la tabla completa y con las indicaciones anteriores, el resultado final sería el siguiente:

La última celda de todas indica cual o cuales son las subsecuencias comunes más largas para las secuencias X = (GAC) e Y = (AGCAT), que en este caso son (AC), (GC) y (GA).

Cuando hay múltiples secuencias y estas son largas, se requieren grandes cantidades de espacio

	0	A	G	C	A	T
0	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
G	$\emptyset$	$\leftarrow^{\uparrow} \emptyset$	$\nwarrow (G)$	$\leftarrow (G)$	$\leftarrow (G)$	$\leftarrow (G)$
A	$\emptyset$	$\nwarrow (A)$	$\leftarrow^{\uparrow} (A)\&(G)$	$\leftarrow^{\uparrow} (A)\&(G)$	$\nwarrow (GA)$	$\leftarrow (GA)$
C	$\emptyset$	$\uparrow (A)$	$\leftarrow^{\uparrow} (A)\&(G)$	$\nwarrow (AC)\&(GC)$	$\leftarrow^{\uparrow} (AC)\&(GC)\&(GA)$	$\leftarrow^{\uparrow} (AC)\&(GC)\&(GA)$

Tabla 2.3: Tabla *LCS* completada

de almacenamiento, este se puede reducir guardando únicamente la longitud más larga de la subsecuencia de cada celda y la dirección de las flechas.

De este modo las subsecuencias pueden obtenerse haciendo una reconstrucción o un **traceback** siguiendo el sentido de las flechas desde la última celda de la tabla (la última celda con el valor más alto). Cuando hay dos flechas en la misma celda de la tabla existen varios caminos posibles. El traceback se detiene cuando se llega a un valor 0 en una de estas celdas. Como vemos a continuación empleando esta técnica se puede reconstruir la subsecuencia (GA) que era una de las obtenidas al terminar de calcular la tabla *LCS* completa.

	0	A	G	C	A	T
0	0	0	0	0	0	0
G	0	$\leftarrow^{\uparrow} 0$	$\nwarrow 1$	$\leftarrow 1$	$\leftarrow 1$	$\leftarrow 1$
A	0	$\nwarrow 1$	$\leftarrow^{\uparrow} 1$	$\leftarrow^{\uparrow} 1$	$\nwarrow 2$	$\leftarrow 2$
C	0	$\uparrow 1$	$\leftarrow^{\uparrow} 1$	$\nwarrow 2$	$\leftarrow^{\uparrow} 2$	$\leftarrow^{\uparrow} 2$

Tabla 2.4: Ejemplo de traceback donde se almacena la *LSC* y el sentido del camino posible

```
1  C ←array[0..m, 0..n]
2  for i ← 0 to m do
3  |  C [i, 0]←0
4  end
5  for j ← 0 to n do
6  |  C [0, j]←0
7  end
8  for i ← 1 to m do
9  |  for j ← 1 to n do
10 |  |  if X [i] = Y [j] then
11 |  |  |  C [i, j]← C [i − 1, j − 1] + 1 else C [i, j]← max ( C [i, j − 1], C [i − 1, j] ) ;
12 |  |  end
13 |  end
14 end
15 return C[m, n]
```

Algoritmo 2.1: Creación de la matriz LCS para dos secuencias de entrada.

Como podemos ver en el Algoritmo 2.1, el pseudocódigo genera la matriz *LCS* para dos secuencias de entrada *X[1...m]* e *Y[1...n]* y obtener la longitud de la subsecuencia, calculando el *LCS X[1...i]* e *Y[1...j]* y almacenando en *X[1...m]* e *C[i,j]*. Al final devuelve la matriz *C[m,n]*, que contiene en cada celda la longitud *LCS* de *X* e *Y*.

2.4. Algoritmo Smith-Waterman

El algoritmo Smith-Waterman (SW) , es el algoritmo principal en el que se basa este proyecto. Este algoritmo es una variante del problema *LCS* que se emplea para realizar alineaciones de secuencias con el objetivo de determinar regiones similares entre dos cadenas de secuencias. En lugar de analizar toda la secuencia, el algoritmo SW compara segmentos de todas las longitudes posibles y optimiza la medida de similitud con el fin de obtener la secuencia común mas larga permitiendo huecos entre ambas.

Esta estrategia es empleada comúnmente en bioinformática, para realizar alineamiento de secuencias biológicas como ADN, ARN o proteínas. Esto se debe a que recientemente, los proyectos de genoma realizados en una gran variedad de organismos, generaron cantidades masivas de datos de secuencia para genes y proteínas que requería de un análisis computacional. La alineación de estas secuencias muestra relaciones entre dichos genes o entre las proteínas, que permite comprender mejor su homología y funcionalidad. La motivación para el uso y desarrollo de este algoritmo radica en la dificultad de encontrar alineaciones correctas en las regiones de muy baja similitud entre secuencias biológicas relacionadas, ya que las mutaciones han agregado demasiado “ruido” a lo largo del tiempo evolutivo. La alineación local evita estas regiones por completo y se enfoca en aquellas con un valor positivo, es decir, aquellas con una señal de similitud que se ha conservado durante el tiempo evolutivo.

En este proyecto aprovecharemos el potencial de este algoritmo para ignorar estas regiones de baja similitud y localizar alineaciones correctas entre la secuencia de visualización canónica de una saga de películas y la secuencia de visualización real del usuario, para así poder estudiar patrones de visualización de los usuarios en las sagas de películas con el objetivo de mejorar los SR de estas.

Este algoritmo garantiza encontrar la alineación local óptima entre dos secuencias empleando una matriz de sustitución y un esquema de puntuación por huecos (en ingles: gap). Del mismo modo que en *LCS* el procedimiento de traceback para generar la subsecuencia comienza en la celda de la matriz de sustitución con el valor más alto y continua hasta encontrar una celda con valor cero, lo que produce la subsecuencia local de mayor valor o más larga.

El algoritmo *SW*, inicialmente tenía un coste computacional de orden ***O(mn)*** para alinear dos secuencias de longitud *m* y *n*. Más adelante se consiguió reducir de ***O(mn)*** a ***O(n)***, es decir a un coste lineal, donde *n* es la longitud de la secuencia más corta de ambas. Esto es en caso de que únicamente se desee una de las múltiples alineaciones óptimas de las secuencias, que es lo que se requiere en

este proyecto.

2.4.1. Detalle del algoritmo SW

El funcionamiento del algoritmo *SW* donde ***A = (a_1,a_2... a_n)*** y ***B = (b_1,b_2... b_m)*** son las secuencias a alinear de tamaño *n* y *m* respectivamente, es el siguiente:

1.- Determinamos la matriz de sustitución y el esquema de penalización por gap.

- ***s(a,b)*** = Valor de similitud de los elementos que constituyen las dos secuencias.
- ***W_k*** = La penalización por gap que tiene longitud ***k***.

2.- Construimos la matriz de puntuación ***H*** e inicializamos la primera fila y la primera columna a cero. El tamaño de la matriz de puntuación es ***(n+1)*(m+1)***.

$$H_{k0} = H_{0l} = 0 \quad for \quad 0 \leq k \leq n \quad and \quad 0 \leq l \leq m$$

3.- Rellenamos la matriz de puntuación empleando la siguiente ecuación:

$$H_{ij} = max \begin{cases} H_{i-1,j-1} + (a_i, b_j), \\ \max_{k \geq 1}\{H_{i-k,j} - W_k\}, \\ \max_{l \geq 1}\{H_{i,j-l} - W_l\}, \\ 0 \end{cases} \quad (1 \leq i \leq n, \leq j \leq m)$$

Donde ***$H_{i-1,j-1}$+s(a_i,b_j)*** es la puntuación de alinear ***a_i*** y ***b_j***.
$H_{i-k,j}$-W_k es la puntuación si ***a_i*** se encuentra al final de un gap de longitud ***k***.
$H_{i,j-l}$-W_l es la puntuación si ***b_j*** se encuentra al final de un gap de longitud ***l***.
0 indica que no existe similitud entre ***a_i*** y ***b_j***.

4.- **Traceback**: Se comienza en la puntuación mas alta de la matriz de puntuación ***H*** y se recorre hasta finalizar en una celda de la matriz con valor 0. El recorrido se realiza tomando el valor más alto de las celdas contiguas hacia la izquierda de la celda actual, generando así la mejor alineación local.

Como podemos apreciar, el algoritmo *SW* alinea dos secuencias por coincidencias/errores, inserciones y eliminaciones. Las operaciones de inserción y eliminación son las que introducen los **gap** o espacios los cuales se representan comúnmente por guiones. Aclaremos ahora cada uno de los pasos definidos anteriormente que conforman el algoritmo.

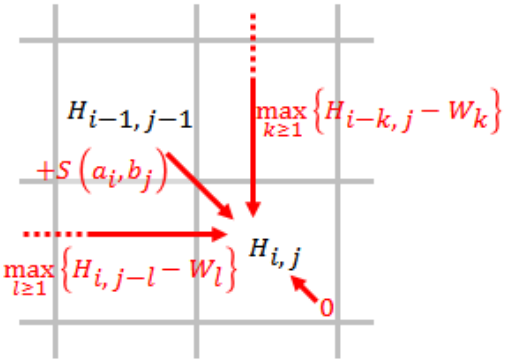

Figura 2.1: Metodo de puntuación del algoritmo Smith-Waterman

Definición de la matriz de sustitución y el esquema de penalización por gap

La matriz de sustitución asigna a cada par de elementos de las subsecuencias, una cierta puntuación si ambos son iguales o si son diferentes. De manera general las coincidencias obtienen puntuaciones positivas, mientras que los errores obtienen puntuaciones más bajas o negativas. La función de penalización por gap determina el coste de abrir o ampliar un gap. Es un algoritmo muy versátil que permite indicar el sistema de puntuación apropiado en función de los objetivos. En este proyecto explotaremos esta propiedad del algoritmo y probaremos diferentes combinaciones de matrices de sustitución y penalizaciones por gap.

Para los siguientes ejemplos definiremos la siguiente matriz de sustitución:

$$s(a_i, b_j)a = \begin{cases} +3, a_i = b_j \\ -3, a_i \neq b_j \end{cases}$$

Si los elementos de la secuencia coinciden se asigna +3 a dicha celda, si no coinciden -3. De este modo la matriz de sustitución quedaría de la siguiente manera:

	A	G	C	T
A	3	-3	-3	-3
G	-3	3	-3	-3
C	-3	-3	3	-3
T	-3	-3	-3	3

Tabla 2.5: Ejemplo de matriz de sustitución

La penalización de **gap** designa las puntuaciones por insertar o eliminar. Una sencilla estrategia es utilizar una puntuación fija para cada gap. En biomedicina, la puntuación debe contarse de manera diferente por razones prácticas pero para todo lo relacionado con este proyecto, asignar un único valor de penalización es una buena estrategia.

En este algoritmo, los huecos o gaps conectados que forman un hueco mayor, son mas favorecidos que múltiples huecos cortos y dispersos. Esto se tiene en cuenta diferenciando los costes por gap de abrir un nuevo hueco o extenderlo. El coste de apertura de un nuevo gap es generalmente mas alto que el coste de extensión de un gap.W_k es la función de penalización por gap para un gap de longitud k

Lineal: La penalización de gap lineal tiene el mismo coste por abrir que por extender un gap, de modo que no existe diferencia entre ambos:
$W_k=kW_1$, donde W_1 es el coste de un gap.

Afín: La penalización de gap afín, considera la apertura o extensión de un gap por separado:
$W_k=uk+v\ (u>0,v>0))$, donde *v* es el coste de apertura de un gap y *u* es el coste de extensión. La penalización, por ejemplo, de un gap de longitud 2 es $2u+v$.

Si tomamos como ejemplo las secuencias TACGGGCCCGCTAC Y TAGCCCTATCGGTCA, las subsecuencias generadas asignando diferentes costes o no a la apertura y extensión de un gap son distintas:

T	A	C	G	G	G	C	C	C	G	C	T	A	-	C
|	|				|		|	|		|	|	|		|
T	A	-	-	-	G	-	C	C	-	C	T	A	-	C

Tabla 2.6: Ejemplo con coste de apertura y extensión de gap de 1

T	A	C	G	G	G	C	C	C	G	C	T	A
|	|				|	|	|			|	|	|
T	A	-	-	-	G	C	C	-	-	C	T	A

Tabla 2.7: Ejemplo con coste de apertura de y extensión de gap de 5 y 1 respectivamente

n En el ejemplo podemos ver cómo la penalización de gap afín, puede ayudaros a evitar pequeños huecos dispersos.

Matriz de puntuación

El objetivo de la matriz de puntuación es realizar las comparaciones individuales entre todos los componentes de dos secuencias y registrar los resultados óptimos de alineación. La alineación óptima actual se obtiene al decidir qué ruta (coincidencia/error o gap) proporciona la puntuación más alta de alineación.

El tamaño de la matriz es la longitud de la primera secuencia más 1 por la longitud de la segunda secuencia más 1. Tanto la primera fila como la primera columna se inicializan completamente a 0 para que el gap inicial no se penalice.

A continuación veamos un ejemplo de cómo rellenar la matriz de puntuación con las dos secuencias TGTTACGG y GGTTGACTA. Utilizaremos las siguiente matriz de sustitución y la siguiente penalización por gap:

- Matriz de sustitución:

$$s(a_i, b_j)a = \begin{cases} +3, a_i = b_j \\ -3, a_i \neq b_j \end{cases}$$

- Penalización por gap: $W_k = 2k$ (una penalización de gap lineal de $W_1 = 2$)

A continuación se muestra el proceso de puntuación de los tres primeros elementos. El color amarillo indica los elementos de la secuencia que se están comparando y el color rojo, la puntuación más alta posible para la celda que se está calificando.

Recordemos que la función empleada para elegir el valor de la celda que se esta comparando es el máximo entre:

$$H_{ij} = max \begin{cases} H_{i-1,j-1} + (a_i, b_j), \\ max_{k \geq 1}\{H_{i-k,j} - W_k\}, \\ max_{l \geq 1}\{H_{i,j-l} - W_l\}, \\ 0 \end{cases} \quad (1 \leq i \leq n, \leq j \leq m)$$

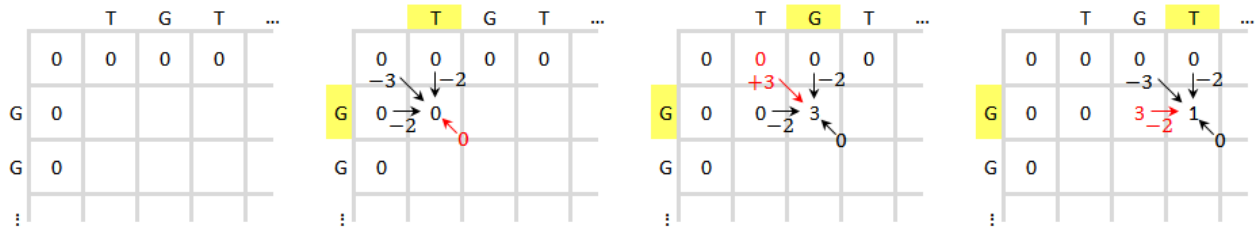

Figura 2.2: Inicializaciń de la matriz de puntuación y del proceso de puntuación de los tres primeros elementos

Traceback

La Figura 2.3 muestra la matriz de puntuación completa. En ella podemos apreciar el proceso de traceback siguiendo la ruta marcada en azul. En el caso de que se tengan varias puntuaciones del mismo valor alrededor de una celda, el rastreo puede realizarse con cada puntuación más alta, dando así lugar a múltiples rutas. También puede indicarse un orden de prioridad por el que se comprueban las celdas colindantes teniendo así una ruta única. Esta segunda opción es la que se ha implementado en este proyecto.

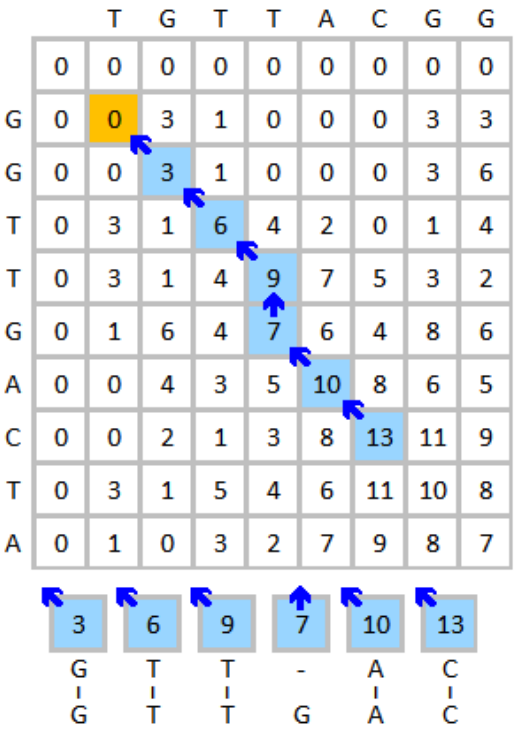

Figura 2.3: Proceso de traceback y alineación final

3

Diseño y Desarrolo

Para el diseño de este proyecto se ha seguido un modelo incremental e iterativo. De este modo se ha dividido el problema principal en diversas etapas de desarrollo con la intención de simplificarlo. Cada una de estas partes cuenta con sus propias pruebas independientes y la modularidad de estas permite ajustarse a las variaciones del proyecto sin afectar al conjunto. A continuación, antes de explicarlo detalladamente, daremos una explicación general de las partes que componen el proyecto en su totalidad y una breve descripción de estas.

El objetivo principal del proyecto, recordemos, que es el estudio de secuencias en un dominio de películas y su aplicación a los sistemas de recomendación aplicando algoritmos de alineación de secuencias sobre conjuntos de datos extraídos de la web.

Para poder lograr este propósito, se ha dividido el proyecto en los siguientes 4 módulos:

1. **Crawler/Scraper:** Este módulo es el encargado de visitar las páginas web de IMDb y descargar el contenido html de estas para procesarlo de manera local. Para obtener dicha información, se transforman los datos en formato HTML, en datos estructurados en ficheros txt que pueden ser utilizados por los demás programas de crawling con el fin de obtener un conjunto de ficheros, que una vez procesados, puedan servir como entrada al algoritmo SW.

3. **Procesamiento de los datos:** Parte dedicada a procesar todos los ficheros obtenidos mediante el proceso de crawling y scraping. Para ello, se agruparán en un formato que incluya toda la información relevante para el algoritmo en ficheros .dat que sirvan como entrada para dicho algoritmo. De esta forma, podremos clasificar e identificar los patrones de visualización de los usuarios recopilados.

2. **Algoritmo Smith-Waterman:** Módulo que contiene la implementación del algoritmo Smith-Waterman explicado en el capítulo 2 aplicado a los datos recopilados previamente con distintos valores de penalización por gap, acierto y error.

4. **Visualización de los datos:** Esta última parte del proyecto esta dedicada a la representaciń gráfica de todos los datos obtenidos aplicando el algoritmo SW a los datos recopilados y procesados previamente de la web. Como resultado final generaremos una serie de gráficas

que reflejan el número de usuarios que visualizan una subsecuencia de películas (algunas películas de la saga), para una determinada secuencia de películas (todas las películas de dicha saga).

3.1. Crawler/Scraper

Como dijimos anteriormente, el objetivo principal del crawler es recopilar información de la web descargando su contenido HTML y una vez descargado, el scraper procesa este formato HTML para obtener la información relevante. Para explicar detalladamente los objetivos de este módulo, se van a definir brevemente los requisitos funcionales y no funcionales de este.

Requisitos funcionales:

RFA01 Recolección de ids de usuarios de IMDb: El sistema permitirá obtener los identificadores de cada uno de los usuarios de ImdB que sean accesibles mediante búsquedas web y almacenarlos en un fichero txt.

RFA02 Extracción de check-ins de los usuarios: El sistema debe poder obtener a partir del id de los usuarios, toda la información asociada a las películas en las que ha hecho check-in (películas que está viendo o que ha visto previamente).

RFA03 Creación de un diccionario de películas: El sistema debe poder descargar, a partir de la lista de check-ins de los usuarios, todo el contenido HTML e información de todas las películas de los usuarios de ImdB.

Requisitos no funcionales:

RNFA01 Portabilidad: El sistema debe poder ser ejecutado correctamente sobre cualquier SO con acceso a internet.

RNFA02 Flexibilidad: El sistema debe funcionar correctamente con diferentes URLs y poder descargar su contenido. Así como permitir implementar de manera sencilla cambios en los datos que se quieran extraer de estas.

RNFA03 Velocidad: El sistema debe ser rápido a la hora de extraer y analizar la información de la web, acorde siempre al número de URLs que visite.

Elección del lenguaje de programación:

En este proyecto se ha decidido implementar el módulo de crawler/scraper utilizando dos lenguajes de programación diferentes. Esto se ha hecho con el objetivo de comparar qué lenguajes son más claros, sencillos y precisos a la hora de extraer información de la web. Por ello, se han escogido los lenguajes de alto nivel de Python y Java para su implementación por los siguientes motivos:

- Ambos son lenguajes de alto nivel orientados a objetos.
- Son lenguajes muy demandados actualmente y con los que se practica a lo largo de la carrera, con lo que son más sencillos de implementar.
- Ambos son lenguajes multiplataforma que permiten un mayor grado de flexibilidad a la hora de ejecutar el proyecto en distintos Sistemas operativos (SO) .

3.1.1. Submódulos

A continuación explicaremos detalladamente cada uno de los submódulos que conforman el módulo crawler/scraper y que permiten cumplir con todos los requisitos definidos previamente.

Recolección de ids de usuarios de IMDb:

Esta parte se ha realizado enteramente en Java, con la intención de comparar el grado de sencillez de un crawler implementado en este lenguaje frente a un crawler implementado en Python. El programa encargado de recopilar los ids de los usuarios de IMDb es imdb_users.java. Este pequeño programa funciona recibiendo como argumento el número de páginas de resultados de búsqueda que se desea analizar. Por cada una de estas páginas se muestran 10 resultados de búsqueda, los cuales contienen la información de los check-ins realizados por el usuario así como su id único.Para ir cambiando entre páginas el programa añade a la URL utilizada como consulta, &start=i, donde i es el número de la página de resultados de búsqueda que se desea analizar. El programa realiza consultas y analiza los 10 resultados de cada una de estas consultas hasta alcanzar el número de páginas indicado como argumento de entrada. Para evitar que Google detecte que el programa es un bot y no una persona realizando consultas normales, se le ha añadido un tiempo de espera aleatorio de entre 0 y 8 segundos para simular un comportamiento humano en las consultas. Cabe destacar que para la obtención de los ids de usuarios, no es necesario descargar el contenido HTML de cada resultado de la búsqueda y aplicarle un proceso de scraper. En el propio HTML de la página de resultados, se encuentra la dirección URL de cada resultado y esta URL ya contiene el id del usuario. Para las consultas y el análisis de cada página de resultados se empleó la librería Jsoup de Java, la cual permite analizar los elementos que hay en una página de resultados de manera trivial. El resultado final es un fichero txt llamado imdb_users, el cual contiene todos los Ids de los usuarios de IMDb.

Extracción de check-ins de los usuarios:

Al contrario que el apartado anterior, este submódulo se ha implementado enteramente en Python. Debido a que el proceso es muy similar al crawling realizado en Java en el submódulo de recolección de ids de usuarios, se quiso probar a implementar un crawler en Python con la intención de comparar el grado de sencillez de ambos. Esta parte es la encargada de obtener toda la información referente a los check-ins realizados por los usuarios de IMDb. El programa encargado de esta funcionalidad es crawling.py. Este programa lee del fichero imdb_users.txt y realiza una conexión HTTP, en inglés

Hypertext Transfer Protocol, con cada uno de los usuarios. Por las mismas razones que en el apartado anterior, se simula el comportamiento de un usuario añadiendo un tiempo de espera aleatorio de entre 1 y 8 segundos para cada una de las peticiones. La diferencia radica en que este programa analiza el contenido HTML de la URL, desplazándose por los diferentes scripts de JavaScript que esta contiene, hasta encontrar en uno de ellos una determinada cadena, la cual se encuentra en todos los HTML de cada usuario, y que contiene un identificador único de una lista con toda la información referida a sus check-ins. El scraping de este submódulo se realizó empleando la libreria BeautifulSoup, la cual permite analizar de manera ágil y dinámica el contenido HTML sin necesidad de descargarlo. Una vez se obtiene ese id se realiza otra petición HTTP a la dirección “https://www.imdb.com/list/” + Id de la lista + “/export”. Con esto obtenemos un fichero .csv que contiene toda la información almacenada en IMDb para cada una de las películas en las que el usuario hizo check-in. De esta manera, el resultado final sería un conjunto de ficheros .csv en el que cada fichero tiene como nombre el id único de la lista de check-ins del usuario.

Creación de un diccionario de películas:

Submódulo encargado de descargar el contenido HTML de todas las películas que el usuario ha visto o está viendo. Para ello se utilizan los ficheros .csv del submódulo anterior. Tras realizar dos submódulos muy similares en dos lenguajes de programación y comparar sus implementaciones, se optó por continuar el proyecto en Python. Esto se debe, principalmente, a que simplifica en gran medida la implementación y el código es más claro y legible para cualquiera que quiera comprender su funcionamiento. De esta manera, este submódulo analiza los ficheros .csv que contienen todos los check-ins que ha realizado el usuario para extraer el id de cada película y realizar una petición HTTP a la URL “https:www/imdb.com/”+ id de la película + “/movieconnections”, para así poder descargar el contenido de manera local y procesarlo detenidamente en el siguiente módulo. De esta manera el resultado final son un conjunto de ficheros HTML con todas las conexiones que tenga una película (películas que la siguen, películas a las que sigue, películas de las que es remake...). Toda la funcionalidad de este submódulo se encuentra en el programa movies_dictionary.py.

3.2. Procesamiento de los datos

El algoritmo SW necesita sequencias de elementos que comparar, es por eso que todos los datos que hemos recopilado mediante el módulo anterior necesitan ser procesados para poder construir una secuencia con los ids de las películas vistas por el usuario (secuencias del usuario) y otra secuencia con los ids de las películas que siguen a una determinada película (secuencias canónicas). Para que este módulo pueda cumplir este propósito, se han definido los requisitos funcionales y no funcionales.

Requisitos funcionales:

RFB01 Creación de secuencias de películas de usuario: El sistema permitirá almacenar en un único documento el id del usuario y los ids de todas las películas en las que este ha hecho check-in, ordenados cronológicamente.

RFB02 Creación de secuencias de películas canónicas: El sistema permitirá almacenar en distintos ficheros el id de una determinada película y los ids de las películas con las que está conectada (películas que la siguen, películas a las que sigue...).

Requisitos no funcionales:

RNFB01 Velocidad: El sistemá debe poder generar estos ficheros de manera rápida, debido al gran tamaño del conjunto de datos a procesar.

Debido a la comodidad y sencillez que el lenguaje de Python demostró en los módulos anteriores, se decidió continuar con este lenguaje para este y para el resto de módulos del sistema. De modo que el lenguaje elegido para el resto del proyecto ha sido Python.

3.2.1. Submódulos

Para poder generar los correspondientes ficheros mencionados en los requisitos este módulo emplea dos submódulos.

Análisis de conexiones entre películas:

Este submódulo tiene como objetivo analizar el contenido de los ficheros HTML descargados en el módulo anterior y generar un fichero con todos los ids de películas y los ids de las películas que están conectadas con esta y el tipo de conexión que tienen (Followed by, Follows, Remade as...). Para ello se analiza su contenido empleando la librería BeautifulSoup, la cual ya hemos utilizado, para buscar los div del documento. Estos divs en el HTML representan la lista de películas con las que está relacionada la película y los títulos de estos, el tipo de conexión que supone esta relación. De este modo, podremos generar un diccionario que incluya como clave los ids de todas las películas y como valor otro diccionario en el que cada clave es el tipo de conexión y el valor todos los ids de las películas que tienen esta conexión con la película que estamos analizando. Con estos datos generamos un fichero llamado movie_connections.dat que incluye todo el contenido de este diccionario.

A continuación generamos para cada tipo de conexión un fichero en el que cada linea contiene el id de la película, seguido de todos los ids de las películas con las que está relacionada separados mediante comas y ordenados. Para ello emplearemos el fichero anteriormente generado de movie_connections.dat y generaremos cuatro nuevos ficheros, uno por cada tipo de relación, llamados; followed_by_sequences.dat, follows_sequences.dat, remade_as_sequences.dat y remake_of_sequences.dat.

De este modo podemos generar una serie de secuencias canónicas, es decir ordenadas por fecha de estreno, para todas las películas de nuestra base de datos. Por ejemplo, si estamos generando el fichero followed_by_sequences.dat para la película con id 0020695, obtendriamos todos los ids de las películas que siguen a 0020695 y en nuestro fichero aparecería la secuencia de películas como: 0020695,0020679,0024210,0025069,0026187,0027656,0144592,0029312. Esta secuencia incluye en primera posición el id de la película seguido de los ids de todas las películas pertenecientes a dicha saga en el orden canónico y separados por comas.

Toda la funcionalidad de este submódulo se incluye en los ficheros test_parse_movieconn.py y movie_sequences.py. El primero permite generar el fichero movie_connections.dat como se indicó anteriormente y el segundo genera los cuatro ficheros con las distintas conexiones.

Creación de secuencias de películas de usuario:

También es necesario obtener una serie de secuencias para los usuarios para poder comparar ambas cadenas en el algoritmo SW (la secuencia canónica y la secuencia del usuario). Para ello utilizaremos todos los archivos csv generados previamente de los check-ins de cada usuario. Como ya vimos cada uno de estos csv contiene en cada una de sus lineas el id de una película que ha visto un determinado usuario y todos estos ids están ordenados cronológicamente (según la fecha en la que hizo check-in de dicha película). De este modo utilizando estos archivos generaremos un archivo llamado user_movies.dat que incluye en cada linea el id de un usuario, un espacio y a continuación todos los ids de las películas que este ha visto separadas por comas y ordenadas cronológicamente.

Toda la funcionalidad de este submódulo se encuentra en el fichero users_sequences.py.

3.3. Algoritmo Smith-Waterman

Este es el módulo más importante del proyecto y el encargado de utilizar todos los datos recopilados y generados por los módulos anteriores. En este módulo se implementará una versión del algoritmo Smith-Waterman similar a la versión definida en el capítulo 2. Esta versión tendrá como objetivo principal el comparar segmentos de todas las longitudes posibles y obtener la subsecuencia común mas larga permitiendo huecos entre la secuencia de visualización canónica de una saga de películas y la secuencia de visualización real del usuario. Para poder lograr implementar de manera eficaz este módulo se han definido los siguientes requisitos funcionales.

Requisitos funcionales:

RFC01 Localización correcta de alineaciones entre dos secuencias de datos: El sistema permitirá realizar alineaciones de secuencias con el objetivo de determinar regiones similares entre dos cadenas de secuencias permitiendo huecos entre ambas y de almacenar

dichas subsecuencias en un fichero de datos.

RFC02 Asignar diferentes valores de acierto o penalización por gap: El sistema permitirá modificar los valores de acierto y la penalización por gap y continuar encontrando la alineación local óptima entre dos secuencias.

Requisitos no funcionales:

RNFC01 Velocidad: El sistema debe poder localizar las subsecuencias y generar estos ficheros de la manera más rapida posible, siendo siempre conscientes del gran volumen de datos a tratar.

RNFC02 Estabilidad: El sistema debe poder ejecutarse de manera estable y soportar grandes periodos de ejecución durante la localización de subsecuencidas debido al gran número de datos a procesar.

Algoritmo Smith-Waterman:

Al contrario que los módulos anteriores, toda la funcionalidad de este se encuentra en un único fichero llamado smith_waterman.py. Este programa contiene una única función que recibe por parámetros; el valor de penalización por gap, el valor de acierto, el valor de error, el fichero de secuencias de usuarios y el fichero de secuencias de películas. Esta función se encarga de aplicar el algoritmo SW, determinando así regiones similares entre las cadenas de secuencias de las películas vistas por los usuarios y las cadenas de secuencias canónicas de las sagas de películas almacenadas.

Para ello analizamos todas las lineas del fichero de secuencias de usuarios y por cada una de estas secuencias realizamos una comparación con cada secuencia de películas canónicas, es decir con cada linea del fichero de películas pasado por parámetros. Para cada una de estas comparaciones generamos la matriz de puntuación para realizar las comparaciones individuales entre todos los componentes de las dos secuencias, estos componentes son los ids de cada película. Inicializamos la matriz de puntuación a 0 y la rellenamos comparando cada uno de los elementos y aplicando la ecuación vista en el segundo capítulo.

Una vez rellenada la matriz aplicamos el proceso de traceback. Para ello, localizamos el valor más alto de la matriz y una vez localizada su posición en esta comenzamos a recorrer el camino de manera inversa hacia el inicio con el fin de obtener la subsecuencia. Como vimos anteriormente en el capítulo 2, el proceso de traceback escoge el valor más alto de las celdas colindantes a la que estamos mirando. Esto puede dar lugar a múltiples rutas de traceback dependiendo de la prioridad que se le dé al análisis de estas celdas, es decir si la primera que miramos es la superior izquierda, la superior o la celda izquierda. El proceso de traceback finaliza cuando encuentra en una de estas celdas el valor 0. Una vez lo encuentra reconstruye la subsecuencia siguiendo todos los pasos desde el valor máximo al primer 0. Para nuestro proyecto hemos obviado estas múltiples rutas y únicamente hemos

generado una. El método de análisis de las celdas adyacentes escogido para nuestra implementación del traceback consiste en analizar primero la celda superior, a continuación la celda izquierda y por último la celda superior izquierda. De estas tres celdas se escoge el valor máximo en dicho orden y finaliza cuando encuentra un 0 en la celda superior izquierda a la celda que estamos analizando en la matriz de puntuación. Se optó por esta implementación debido a los resultados de diferentes pruebas que comentaremos detenidamente en el siguiente capítulo.

De este modo una vez se construyen todas las subsecuencias de todos los usuarios con todas las sagas de películas, almacenamos dicha subsecuencia, con el número de huecos, en un fichero de texto que utilizaremos en los siguientes submódulos. Como dijimos anteriormente toda la funcionalidad necesaria para cumplir los requisitos de este módulo esta implementada en un único fichero que incluye la función de SW y un archivo main.py que llama a esta función con distintos valores de penalización por gap y de acierto y genera los ficheros con dichas subsecuencias.

3.4. Visualización de los datos

Este es el último módulo en el que se divide el proyecto y el encargado de dar una representación gráfica de los resultados obtenidos a lo largo de todo el proceso de recogida y procesamiento de los datos. En este módulo utilizaremos todos los ficheros generados por el módulo anterior para poder obtener una serie de gráficas que nos permitan poder analizar, estudiar y comprender los resultados obtenidos. Para poder generar dichas gráficas y poder interpretar los resultados se han definido los siguientes requisitos funcionales y no funcionales.

Requisitos funcionales:

RFD01 Creación de gráficas: El sistema deberá generar correctamente gráficas en cuyo eje Y se muestre el número de usuarios que hay dada una determinada subsecuencia de tamaño n para cada uno de los diferentes tamaños de sagas de películas (longitud de la secuencia de películas de una determinada saga) indicados en el eje X.

RFD02 Apreciación de los distintos tamaños de subsecuencias y puntos de cambio en la gráfica: El sistema permitirá diferenciar de manera clara y concisa los diferentes tamaños de subsecuencias, así como indicar mediante una leyenda cuales son los distintos tamaños de subsecuencias. Del mismo modo se deberá indicar mediante marcadores los puntos donde para determinado tamaño de subsecuencia, el valor del número de usuarios dado un tamaño de secuencia de películas cambie.

Requisitos no funcionales:

RNFD01 Flexibilidad: El sistema deberá poder generar gráficas de manera correcta y legible independientemente del volumen de datos del fichero y de la distribución de estos.

RNFD02 Portabilidad: El sistema deberá generar gráficas en un formato que sea compatible con los principales editores de texto (Latex y Word).

Generador de gráficas:

Del mismo modo que el módulo anterior, toda la implementación de este se encuentra en un único fichero python llamado plot_gen.py. Dicho fichero incluye una función que emplea como parámetro los ficheros generados por el módulo anterior, con los que genera las gráficas definidas anteriormente en los requisitos funcionales. Para la implementación de esta función se han empleado una serie de librerías externas de Python, las cuáles permiten simplificar mucho el código. Estas librerías son: pandas, seaborn y matplotlib. La librería de pandas es una potente herramienta de Python para tratar y analizar de manera sencilla estructuras de datos. De este modo podemos fragmentar y acceder de manera simple a los datos almacenados en los ficheros generados por el algoritmo SW y separarlos por columnas que faciliten su procesamiento. Matplotlib, es una conocida librería de Python que permite diseñar gráficas en 2D y que es bastante simple e intuitiva. Por último seaborn es una librería Python basada en matplotlib que nos aporta una interfaz más completa, atractiva y de mayor nivel de detalle para representar información estadística de manera gráfica. Esta librería es la que nos permite distinguir de manera sencilla las distintas variables de la gráfica y los puntos donde estas van cambiando su valor.

Empleando estas librerías, hemos realizado una función con la que generamos una serie de gráficas que muestran en su eje X el número de usuarios y en el eje Y el tamaño de las secuencias de las películas canónicas (sagas de películas en orden). Los datos de la gráfica indican las subsecuencias de un determinado tamaño. De este modo para un tamaño de secuencia de películas de n y una subsecuencia de tamaño m, tendremos un número z de usuarios. Por ejemplo para secuencias de películas de tamaño 5, puede haber 2000 usuarios que tengan una subsecuencia común de tamaño 4 entre las películas del usuario y las sagas de películas de tamaño 5.

Diagrama de proceso y comunicación entre los módulos:

Debido a que este proyecto no es una aplicación estructurada que siga un determinado diagrama de pasos, hemos realizado un breve esquema, el cual podemos ver en la Figura 3.1, que resume la comunicación y dependencias existentes entre los distintos módulos comentados anteriormente en este capítulo y el flujo que siguen los datos.

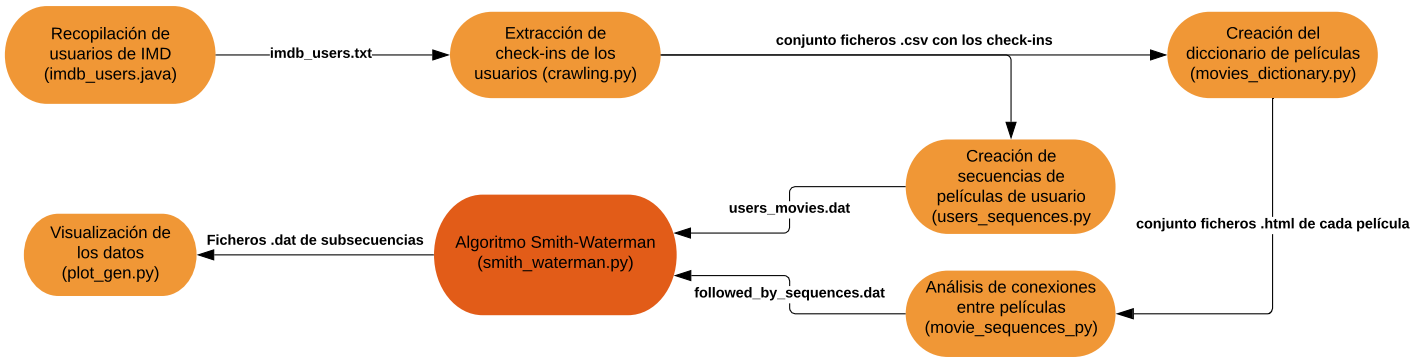

Figura 3.1: Diagrama de proceso y comunicación entre los distintos módulos

4

Pruebas y resultados

Como hemos mencionado anteriormente, una vez aplicado el algoritmo Smith-Waterman sobre el conjunto de datos, se han obtenido una serie de ficheros que contienen todas las subsecuencias comunes existentes entre las películas vistas por los usuarios y las películas pertenecientes a una determinada saga en orden canónico. En este capítulo nos centraremos en analizar estos resultados y mostrarlos de manera gráfica para entender mejor los patrones de visualización de películas de los usuarios y ofrecer modelos de datos con los que sea posible plantear contrastes de hipótesis que puedan mejorar la recomendación de películas pertenecientes a una saga a dichos usuarios.

Experimentos realizados

Con el objetivo de poder recopilar la mayor cantidad de información posible, se ha aplicado el algoritmo SW variando los pesos de acierto, error y penalización por gap. De esta manera podremos variar el tamaño de las subsecuencias comunes en función del número de huecos mediante las distintas penalizaciones por gap.

Para los distintos experimentos se escogieron las conexiones de películas del tipo Followed By como secuencias de películas canónicas a comparar con las secuencias de películas de los usuarios. Las conexiones Followed By contienen el Id de una película y los Ids de las películas que la siguen de manera ordenada. Debido al gran número de pruebas a realizar nos pareció mas interesante centrarnos en este tipo de conexión ya que nos permite generar subsecuencias de películas ordenadas, las cuales nos aportan más información útil a la hora de analizar los patrones de visualización de los usuarios que otros tipos de conexiones como los remakes o los remades, los cuales no se espera que los usuarios vean en orden.

Datasets

En un principio se utilizaron únicamente los datos públicos recopilados de la web mediante crawling de los usuarios de IMDb pero, como veremos a continuación, estos datos no eran suficientes para obtener una información precisa sobre los patrones de orden de visualización de los usuarios. Por esta razón utilizamos también un dataset público del sitio web MovieLens. Este sitio web se centra

en la recomendación automática de películas a los usuarios y posee registros de cientos de miles de usuarios con el objetivo de mejorar los sistemas de recomendación de películas. El dataset de MovieLens contiene todas las películas que ha visto un usuario y el identificador de dicha película en IMDb, con lo que hemos podido adaptar fácilmente los programas empleados en el procesamiento de los datos de usuario obtenidos mediante crawling para que también obtengan la información de las distintas películas de este dataset.

4.1. Análisis de resultados

Inicialmente utilizamos nuestro dataset de IMDb, el cual dispone de alrededor de unos 160 usuarios, y lo comparamos con la secuencia canónica de las películas que tienen una relación de Followed By.

Para las distintas pruebas hemos variado los distintos pesos de penalización por gap, acierto y error a (0,1,-1), (0,1,0), (-2,1,-1) y (-1,3,-3) respectivamente y hemos comprobado el número de usuarios que tienen, entre su secuencia de películas y las secuencias de películas canónicas, una subsecuencia de tamaño n en función de los distintos tamaños de las sagas. Debido a que se han buscado subsecuencias de todos los tamaños, hemos fragmentado algunas gráficas en varias partes con el objetivo de evitar el solapamiento y esclarecer los resultados.

IMDb Followed By 0,1,0

Como se puede apreciar en estos resultados, para los pesos (0,1,0), el algoritmo SW encontrará siempre la subsecuencia común entre la secuencia de películas del usuario y la secuencia de películas de una determinada saga, en el caso de que exista, independientemente del número de gaps que haya entre ambas. En consecuencia, el número de usuarios con subsecuencias comunes será mayor que con el resto de pesos para cualquier tamaño de subsecuencia. Esto se debe a que al no tener ningún tipo de penalización por gap o por error, no importa la longitud del gap entre un valor de la secuencia y el mismo valor en la otra, si están en orden, el algoritmo SW generará la subsecuencia común entre estos. De este modo, con estos pesos se generará siempre la subsecuencia común más largas entre dos secuencias, es decir, es equivalente al valor de LCS. Esto se puede apreciar al ver que para las subsecuencias de tamaño 3, a partir de sagas de más de 27 películas, todos los usuarios han visto al menos 3 de ellas en orden.

De este modo, si estamos interesados en ver si un usuario ha visto las películas de una determinada saga en orden sin importar el tiempo que ha pasado (que en nuestro caso lo tomamos como el número de películas que ha visto entre medias) entre una película y la siguiente de la saga, estos pesos son idóneos para nuestro algoritmo.

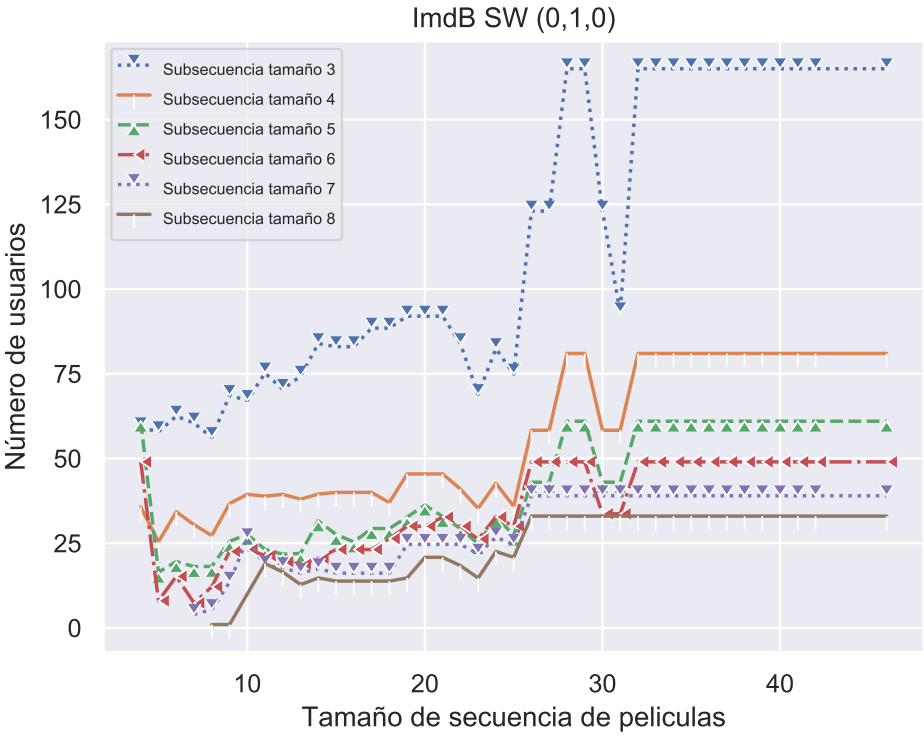

Figura 4.1: Número de usuarios IMDb con secuencias de tamaño 3-8 con (0,1,0)

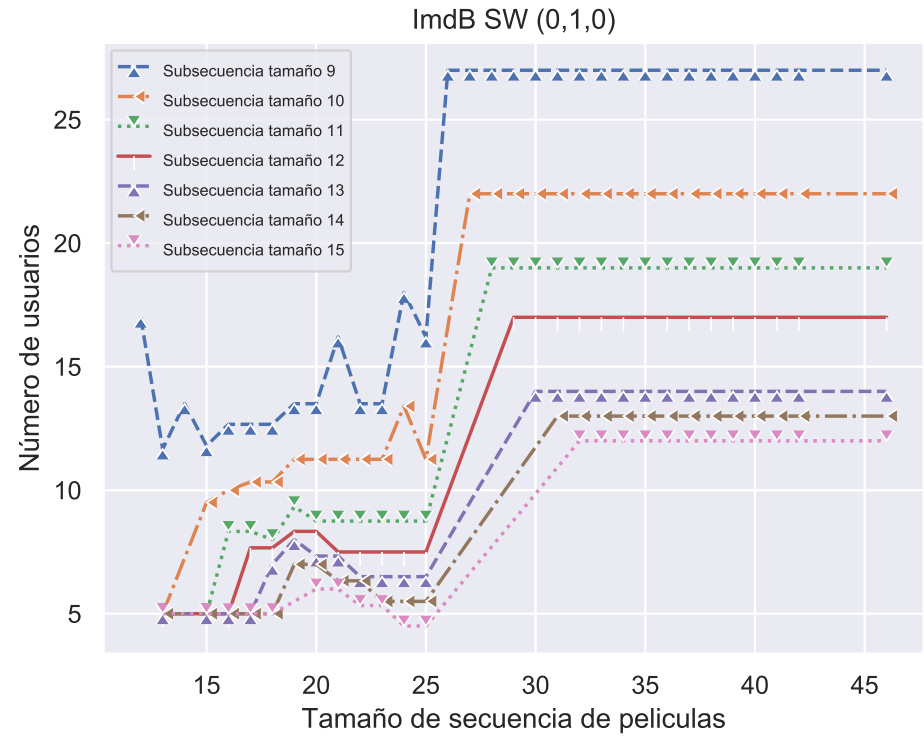

Figura 4.2: Número de usuarios IMDb con secuencias de tamaño 9-15 con (0,1,0)

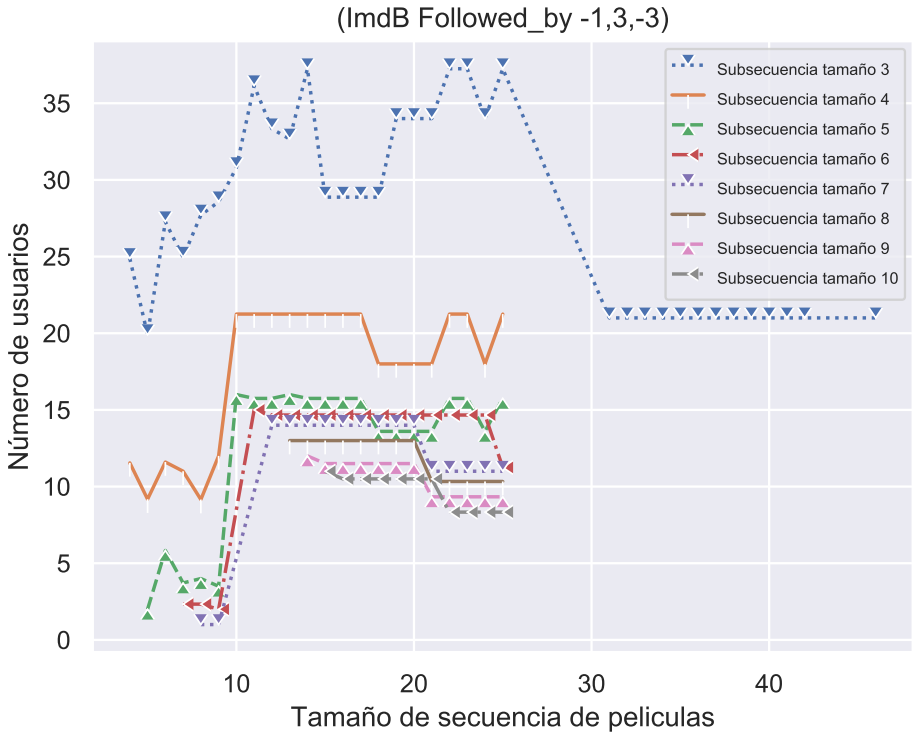

Figura 4.3: Número de usuarios IMDb con secuencias de tamaño 3-10 con (-1,3,-3)

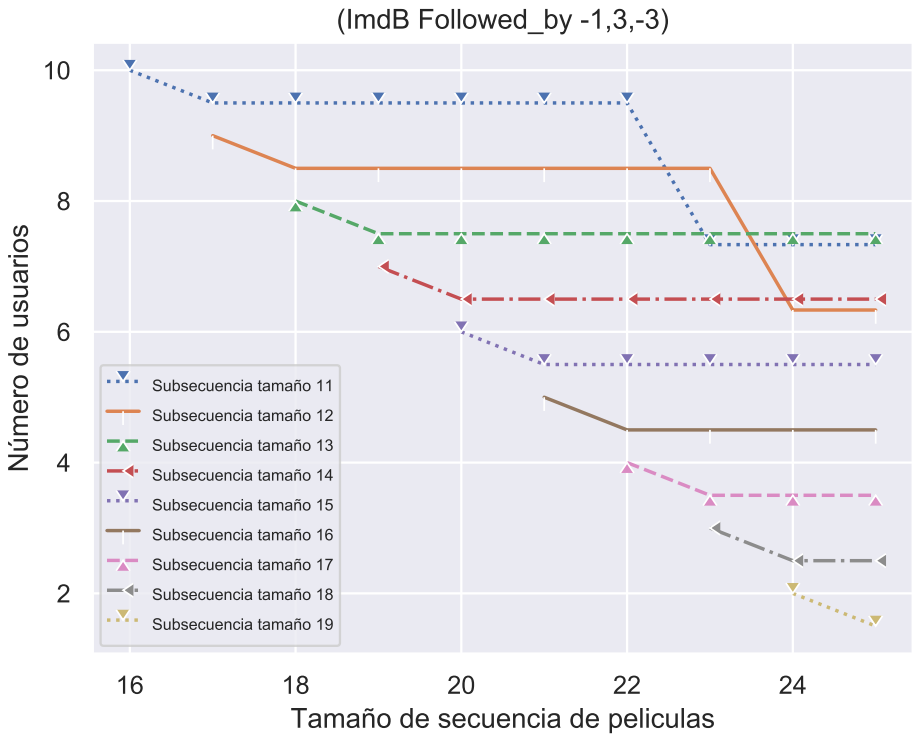

Figura 4.4: Número de usuarios IMDb con secuencias de tamaño 11-19 con (-1,3,-3)

IMDb Followed By -1,3,-3

Para los resultados del algoritmo con pesos (-1,3-3), los resultados son los que se muestran en las Figuras 4.3 y 4.4:

Se puede apreciar claramente un descenso en el número de usuarios con subsecuencias en las sagas de distinto tamaño con respecto a los resultados anteriores con pesos (0,1,0). Esto es así, ya que aquí se introduce una penalización por gap de -1. Añadir una penalización por gap hace más restrictivo el algoritmo, reduciendo el número de subsecuencias comunes y la longitud de estas. Esto se debe a que al añadir una penalización toma importancia el número de gaps que existe entre el valor de una secuencia y el mismo valor en la otra, pudiendo así quedar descartada una subsecuencia común de una longitud mayor debido al gran número de gaps que hay entre esos dos valores. Como se puede ver si comparamos los resultados con estos pesos frente a los resultados anteriores, el mayor número de usuarios con al menos una subsecuencia de tamaño 3 es 37 frente a los 160 en la prueba anterior.

Además, podemos apreciar que en este caso, el máximo de 37 se alcanza en sagas con un tamaño de entre 15 y 20 películas, para luego descender. En el caso anterior este valor máximo se alcanzaba con sagas de mayor tamaño y era ascendente. Esto era porque en sagas tan grandes, el número de gaps entre un acierto y otro al comparar dos secuencias es mayor y al no tener penalización el algoritmo lo contaba como una subsecuencia válida. Por el contrario si se añade aunque sea una penalización de -1, muchas de estas subsecuencias comunes quedarán descartadas.

IMDb Followed By -2,1,-1

En este caso hemos aplicado un factor de penalización por gap de -2, es decir penalizando el doble cada gap con respecto a la prueba anterior. Como podemos apreciar en los resultados, las gráficas son similares en cierto sentido, con la gran diferencia de que el número máximo de usuarios para subsecuencias de tamaño 3 se reduce a poco más de la mitad con respecto a la penalización por gap de -1. Esto se debe a lo comentado anteriormente, sin embargo, no solo se ha reducido el número de los usuarios, también ha variado la distribución de estos. La razón de esto es que no solamente hemos cambiado la penalización por gap, también hemos cambiado los pesos de acierto y de error. Esto hace que las subsecuencias comunes varíen su resultado. Cuanto más peso se le asigna al acierto y al error más probable es que al hacer el proceso de traceback en la matriz de puntuación, el algoritmo encuentre antes un valor cuyas celdas colindantes valgan todas 0. En este caso el algoritmo se detendrá ahí y devolverá una subsecuencia más similar al final, de la secuencia más larga de las que se están comparando, que al inicio de esta. Al asignar un peso de 1 y -1 al acierto y error respectivamente, es probable que el valor más alto de la matriz esté más cerca del inicio de esta que con unos pesos de 3 y -3. De este modo al realizar el proceso de traceback es más probable, aunque no siempre, que el primer 0 que encuentre sea el 0 inicial de la matriz de puntuación.

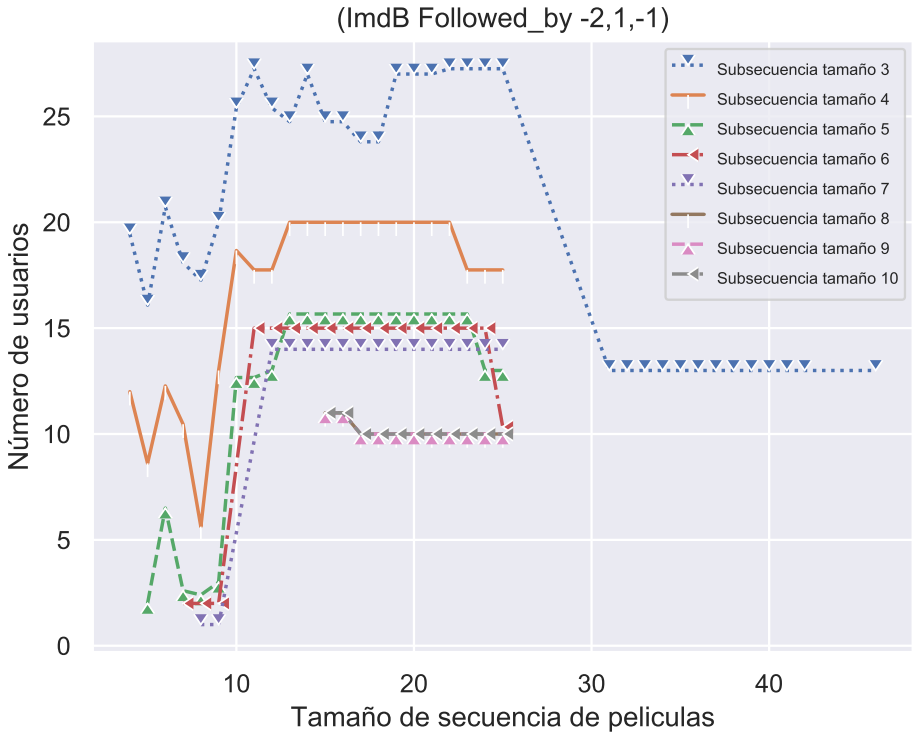

Figura 4.5: Número de usuarios IMDb con secuencias de tamaño 3-10 con (-2,1,-1)

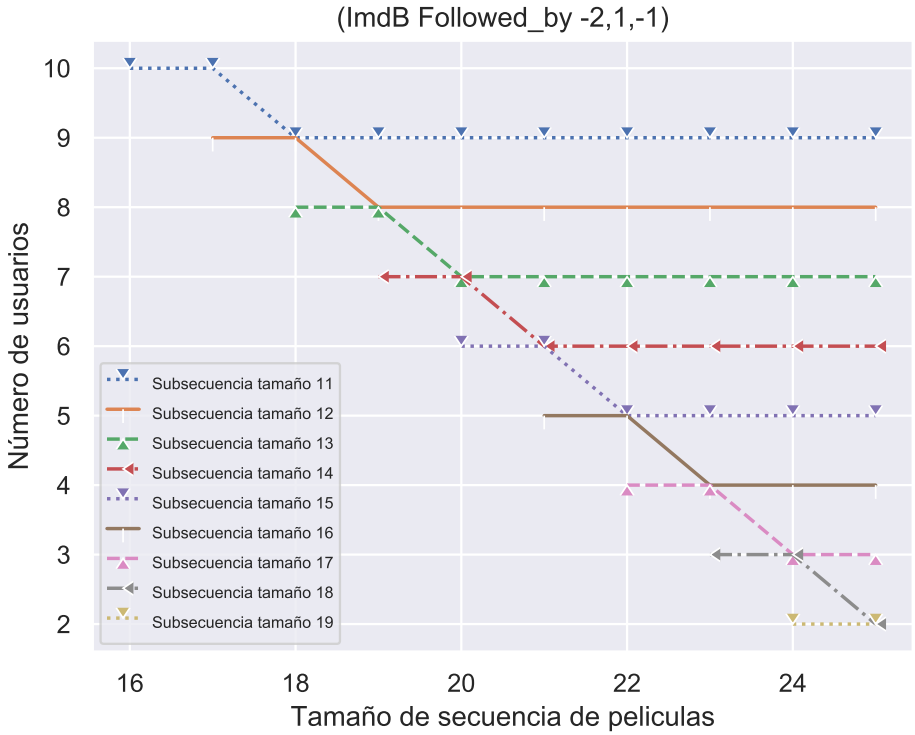

Figura 4.6: Número de usuarios IMDb con secuencias de tamaño 11-19 con (-2,1,-1)

En definitiva podemos concluir que lo que más afecta en la comparación de este tipo de secuencias es la penalización por gap, haciendo que variarla influya en el número de películas que permitimos al usuario ver entre una película que sea la misma en las dos secuencias a comparar y la siguiente, ya que estas películas que no son aciertos en la comparación, son los gaps o huecos de nuestra subsecuencia común. De este modo variando este valor y teniendo en cuenta que podemos tomar el número de películas que el usuario ve entre medias como el tiempo que tarda en ver una película y su continuación, se podría medir, entre otras cosas, el tiempo que tarda un usuario en seguir una saga que ha visto completamente en función del número de gaps.

Aun así y habiendo obtenido una serie de resultados que nos muestran cuántos usuarios de 160 poseen subsecuencias comunes entre las películas que han visto y las sagas a las que estas pertenecen, es cierto que 160 usuarios no es un número lo suficientemente alto como para poder sacar conclusiones consistentes que respalden alguna hipótesis. Esto se debe a que es probable que muchos de estos usuarios tengan unos gustos muy similares, con lo que el conjunto de datos no aportaría demasiada información. Otro problema es que los datos de los usuarios son recogidos sobre los check-ins que estos han realizado en algunas películas y un usuario puede hacer check-in sobre una película que aun no ha visto y únicamente está interesado en ver, al contrario de los ratings que son más fiables en este sentido. Por ello se decidió aplicar el algoritmo SW con los mismos pesos en el conjunto de datos de MovieLens mencionado al inicio de este capítulo. Estos datos incluyen información sobre 10200 usuarios y los Ids de IMDb de las películas que estos han puntuado. Este dataset ha sido recopilados por GroupLens, el cual es un departamento de investigación de ingeniería informática y ciencia de datos de la Universidad de Minnesota.

MovieLens Followed By 0,1,0

Estos resultados muestran la aplicación del algoritmo SW con una penalización por gap de 0 y un peso de acierto y error de 1 y 0 respectivamente. Del mismo modo que en el experimento sobre el conjunto de datos de IMDb con los mismos pesos, estos son los pesos que muestran un mayor número de usuarios con subsecuencias comunes. Esto se debe a que, como comentamos anteriormente, con una penalización por gap de 0, siempre formará, si existe, una subsecuencia común entre la secuencia de películas vistas por el usuario y la secuencia de la saga independientemente del número de huecos que haya entre estos valores de acierto.

Como puede apreciarse en los resultados con el dataset de MovieLens, se puede ver claramente una distribución muy distinta de los datos con respecto al conjunto de datos de IMDb representado en la Figura 4.1. Al aplicar el algoritmo SW con los pesos (0,1,0) en el conjunto de IMDb, obtuvimos que el mayor número de usuarios con una subsecuencia común se daba con subsecuencias de tamaño 3 e iba aumentando a medida que aumentaba el tamaño de la saga de películas. Por el contrario, como podemos ver en la Figura 4.7, en un dataset con mayor número de usuarios y que basa sus datos en ratings, el mayor número de usuarios con una subsecuencia común de tamaño 3 se da cuando el

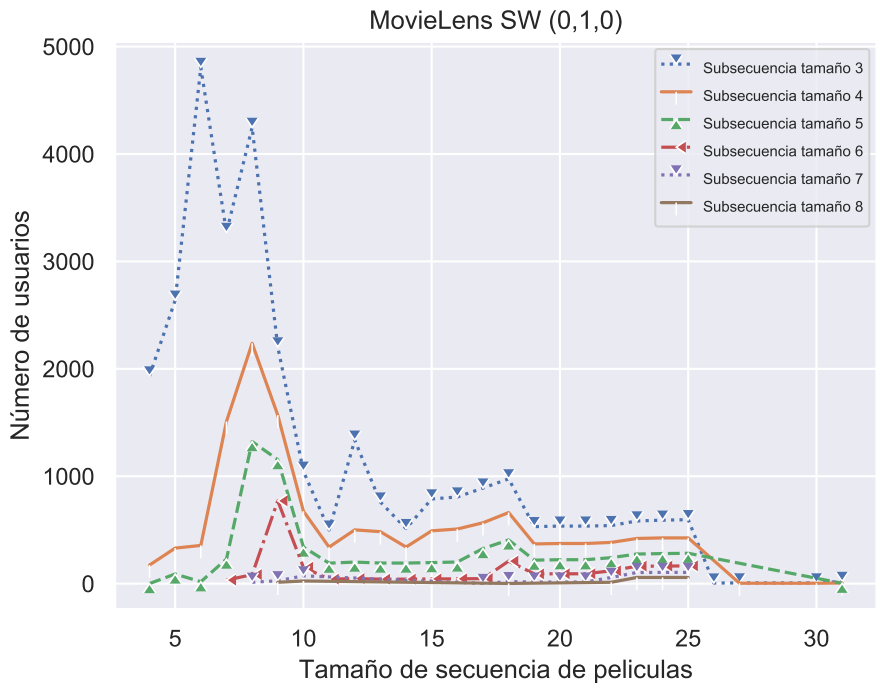

Figura 4.7: Número de usuarios MovieLens con secuencias de tamaño 3-8 con (0,1,0)

tamaño de la saga es de 7 películas. La explicación de esto es que cuando los datos se basan en ratings, son películas que el usuario ha visto y ha puntuado pero cuando los datos se basan en check-ins, puede que el usuario las haya visto o simplemente las haya marcado como que está interesado en ver. Por esta razón es mucho más común que un usuario marque más películas para ver en sagas de hasta 25 películas y que luego no las vea que, el que después las vea realmente. Por ejemplo, en la Figura 4.1 se puede apreciar que el pico más alto del número de usuarios se da en sagas de 25 películas para subsecuencias de películas de tamaño 3 mientras que, en la Figura 4.7, para este valor únicamente hay unos 600 usuarios de 10200. Esta es una de las razones principales por la que se decidió emplear también el conjunto de datos de MovieLens, ademas de poder así comparar ejecuciones del algoritmo SW con los mismos pesos en distintos conjuntos de datos.

Un dato interesante que podemos apreciar de esta gráfica es que las sagas de 3 películas son las sagas que más usuarios ven en orden ya que estas sagas son las que tienen el mayor número de usuarios con subsecuencias de tamaño 3. La explicación más simple es que esta longitud de saga es la más común, se suelen ver seguidas en poco tiempo y que la mayoría de las películas pertenecientes a sagas de este tamaño se estrenan de manera consecutiva en, relativamente, un periodo corto de tiempo.

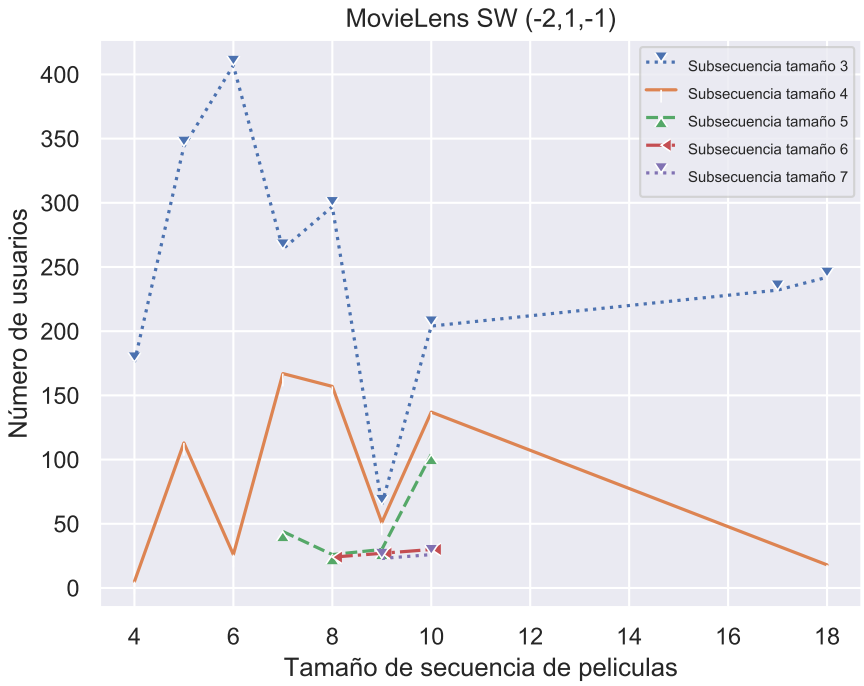

Figura 4.8: Número de usuarios MovieLens con secuencias de tamaño 3-7 con (-2,1-1)

MovieLens Followed By -2,1-1

La ejecución del algoritmo con una penalización por gap de -2 y un peso de acierto y error de 1 y -1 en el conjunto de datos de MovieLens, muestra como resultado la gráfica de la Figura 4.8. En esta gráfica podemos apreciar claramente un descenso del número de usuarios que poseen una subsecuencia común frente a la Figura 4.7 con unos pesos de (0,1,0). El mayor número de usuarios con una subsecuencia común sigue siendo el de las subsecuencias de tamaño 3, con lo que queda claro que estas son las subsecuencias más comunes. Pero a diferencia del número de usuarios anterior, aquí únicamente 400 de los 10200 usuarios tienen al menos una subsecuencia de tamaño 3, el cual es un valor muy inferior a los casi 5000 usuarios de la Figura 4.7. La razón de esto es la misma que comentamos anteriormente. Al añadir una penalización por gap se penaliza el número de huecos que exista entre un acierto en ambas secuencias, descartando así algunas subsecuencias comunes que el algoritmo no descartaría si no hubiese tantos huecos entre cada uno de los valores comunes en las dos secuencias.

Como ya dijimos, el número de huecos o de valores erróneos entre dos secuencias se puede relacionar con el tiempo que un usuario tarda en ver una película y su continuación. No es un aproximación absolutamente cierta, ya que algunos usuarios ven muchas películas de manera regular. Aun así el dataset de MovieLens contiene, en su mayoría, información de usuarios con un número normal de películas y algunos usuarios más específicos con un gran número de películas puntuadas. Por esta razón podemos decir que el número de películas entre medias de una película perteneciente a una

saga y su continuación se puede asociar al tiempo que ha pasado entre que este ha visto las dos. De este modo podemos ver que, en promedio, las sagas que poseen entre 4 y 7 películas son las sagas de las cuales un mayor número de usuarios ha visto al menos 3 películas en orden. Esto indica que estas sagas son las que estrenan sus películas en un menor espacio de tiempo. Un ejemplo claro que podemos ver en la Figura 4.7 es que la saga más larga con mayor número de usuarios para una subsecuencia son las sagas de tamaño 6 con subsecuencias de tamaño 3. Un ejemplo que explica este caso concreto es que un gran número de usuarios han visto de manera ordenada las películas de El Señor de los Anillos. Estas películas se estrenaron de manera consecutiva con un año de diferencia entre ellas y el tamaño canónico de la saga es de 6 películas ya que a dicha saga pertenecen las 3 últimas películas de El Hobbit. De este modo para esta saga de 6, la mayoría de usuarios han visto de manera ordenada 3 de estas películas, que probablemente sean las 3 de El Señor de los Anillos. Por lo tanto se podría decir que una buena estrategia para facilitar que las personas vean de manera ordenada las películas pertenecientes a una saga, es estrenarlas en periodos cortos de tiempo.

MovieLens Followed By -1,3,-3

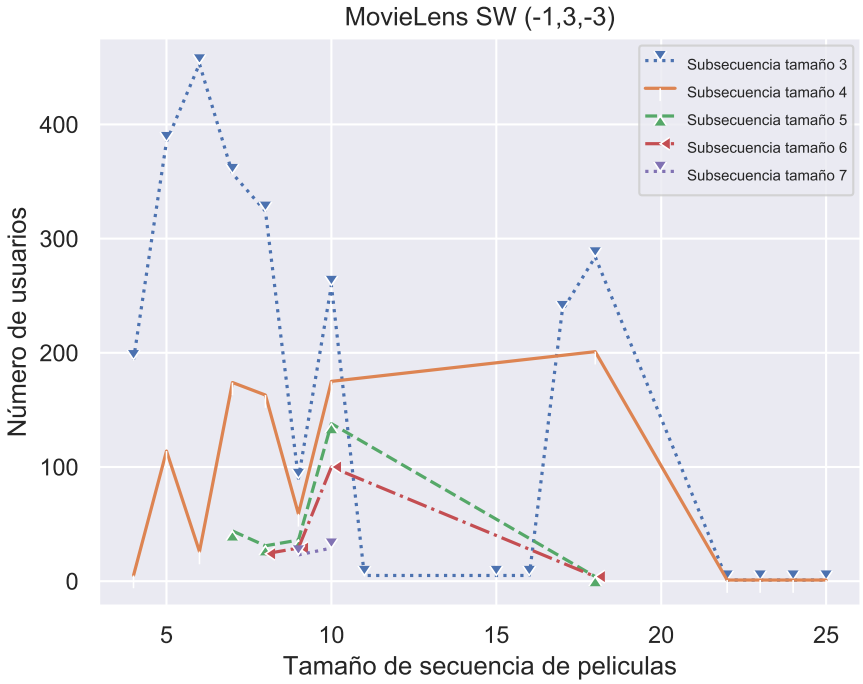

Figura 4.9: Número de usuarios MovieLens con secuencias de tamaño 3-7 con (-1,3,-3)

La última prueba realizada sobre el conjunto de datos de MovieLens fue la realizada aplicando una penalización de -1 a los gaps en lugar de -2 y un peso de acierto y error de 3 y -3. Al aplicar la mitad de penalización que en el caso anterior, en la Figura 4.9, puede apreciarse que se incrementa el número de usuarios con subsecuencias comunes para algunos tamaños de subsecuencia aunque para otros se mantiene igual. Lo que si que es una diferencia claramente apreciable entre ambas gráficas es que

las dos siguen un comportamiento similar hasta las sagas de tamaño 10 y que a continuación cambia completamente. Como podemos ver en la Figura 4.8, para la penalización de gap de -2, las sagas de tamaño 10 en adelante, no poseen ninguna subsecuencia común para subsecuencias de más de 4 y para las de tamaño 3 y 4 se va reduciendo de manera gradual el número de usuarios. Sin embargo con una penalización de -1, para las sagas de tamaño 10 o más, el número de subsecuencias comunes para subsecuencias mayores de 4 se reduce de manera regular en lugar de desaparecer (a excepción de las subsecuencias de tamaño 7). Por el contrario, para las subsecuencias de tamaño 4, crece de manera regular hasta las sagas de 19 películas y luego decrece. La razón principal es que al tener la mitad de penalización, se dan por válidas una cierta cantidad de subsecuencias que tenían un mayor número de huecos a partir de las sagas de tamaño 10, las cuales se van reduciendo gradualmente, y que en el caso de la penalización de -2 se descartaban directamente.

El caso particular se da con las subsecuencias de tamaño 3. En ambos casos hay un valor alto de usuarios con subsecuencias comunes de este tamaño en las sagas de 18 películas. La diferencia reside en que con una penalización por gap de -2, el número de usuarios se mantiene creciente hasta dicho punto y con una penalización de -1, decrece a partir de las sagas de 10 películas hasta casi 0 para volver a crecer bruscamente a partir de las sagas de 16. Este es un comportamiento atípico que no se debe al peso de la penalización por gap si no a los pesos de acierto y error. Cuando se asignan unos pesos elevados de acierto y error, como explicamos anteriormente al aplicar los pesos (-1,3,-3) en el conjunto de IMDb, es más probable que al recorrer la matriz de puntuación en el proceso de traceback, el algoritmo encuentre antes un valor cuyas celdas colindantes valgan todas 0. Si por algún casual esto sucede al empezar el proceso de traceback, se producirán subsecuencias comunes de longitud 1 o 2. Estas subsecuencias quedarán descartadas y se producirán resultados atípicos en nuestra gráfica. Por esto es importante calcular bien los pesos de acierto y error en el algoritmo SW y probar diferentes pesos que no generen resultados anómalos.

4.2. Distribución de los datos

Además de la representación gráfica de los resultados y del estudio de estos, existe otro tipo de análisis matemático que nos permitirá poder plantear hipótesis sobre el conjunto de datos y verificarlas o refutarlas en función de los datos. Este tipo de análisis es el estudio de la distribución de probabilidad, el cual nos permite ver si los resultados se ajustan a un cierto modelo probabilístico. Esto es de suma importancia para analizar los patrones de visualización de los usuarios ya que nos permitirá poder realizar contrastes de hipótesis que respondan a las preguntas que planteemos sobre sobre los resultados.

Para estudiar la distribución de probabilidad, primero tenemos que elegir un conjunto de resultados que analizar. Basándonos en los resultados obtenidos anteriormente tras la aplicación del algoritmo SW con diferentes pesos y sobre los distintos dataset, concluimos que el más idóneo para analizar su

distribución probabilística era el conjunto de resultados del dataset de MovieLens con los pesos (0,1,0). Esto se debe a que el conjunto de MovieLens muestra datos más reales al estar basado en puntuación por ratings y la aplicación del algoritmo SW con esos pesos muestra todas las subsecuencias comunes entre los usuarios y las sagas independientemente del número de películas no pertenecientes a la saga que hayan visto entre medias. De este modo, el volumen de datos de estos resultados es el de mayor tamaño y el más próximo al comportamiento real de los usuarios que tenemos.

La distribución de probabilidad es una función que asigna a cada suceso definido sobre una variable aleatoria la probabilidad de que dicho suceso ocurra. De este modo lo primero que tenemos que hacer es definir la variable aleatoria que queramos estudiar en el conjunto de datos. En nuestro caso hemos definido esta variable aleatoria como el número de subsecuencias de tamaño n que hay por cada saga de tamaño m. Esto, lo que realmente nos indica, es cuantos usuarios han visto para una determinada saga de tamaño m, al menos n películas en orden.

Para ello hemos modificado un poco el fichero de datos obtenido tras aplicar SW (0,1,0) a MovieLens, para poder adaptarlo como argumento de entrada a un programa encargado de buscar el modelo probabilístico que mejor se adapta a la distribución de los datos. En nuestro caso únicamente lo probamos para subsecuencias de tamaño 3 y 4, las cuales creemos, en base a los resultados obtenidos, que son las más representativas y las que más información pueden aportar.

Los resultados que podemos ver en las Figuras 4.10 y 4.11 muestran la distribución de esos datos y el mejor ajuste posible a dicha distribución.

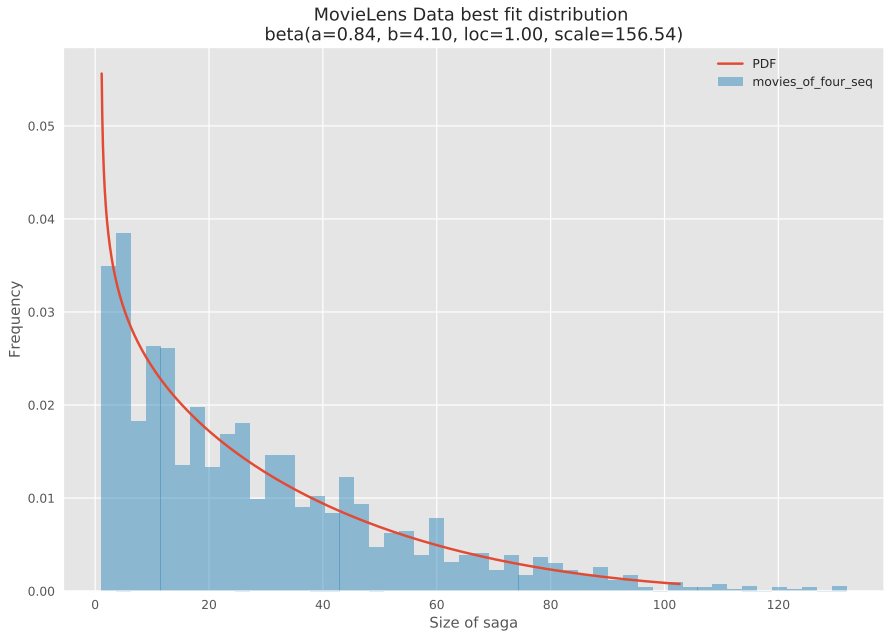

Figura 4.10: Número de subsecuencias de tamaño 3 vistas por los usuarios para sagas de distinto tamaño

En la Figura 4.10 podemos ver que el valor más alto se alcanza para usuarios que solamente han visto 7 sagas de películas de tamaño 3. Esto, si lo analizamos, tiene bastante sentido ya que en la historia del cine, el número de sagas de 3 películas más famosas, y que los usuarios hayan visto con mayor probabilidad, puede que ronde en unas 20 sagas (Regreso al futuro, Blade, El caso Bourne, etc). Por lo tanto, el que la mayoría de usuarios hayan visto entre 6 y 8 de dichas sagas en orden, es una posibilidad razonable.

Aparte de este detalle, se puede apreciar claramente que la distribución de los datos es decreciente y que a medida que aumenta el número de sagas de películas de tamaño 3 que el usuario ha visto, la frecuencia del número de dichas sagas que ha visto en orden se reduce. El modelo probabilístico que mejor se adapta a estos resultados, es la distribución beta con $\alpha = 0{,}84$ y $\beta = 4{,}1$.

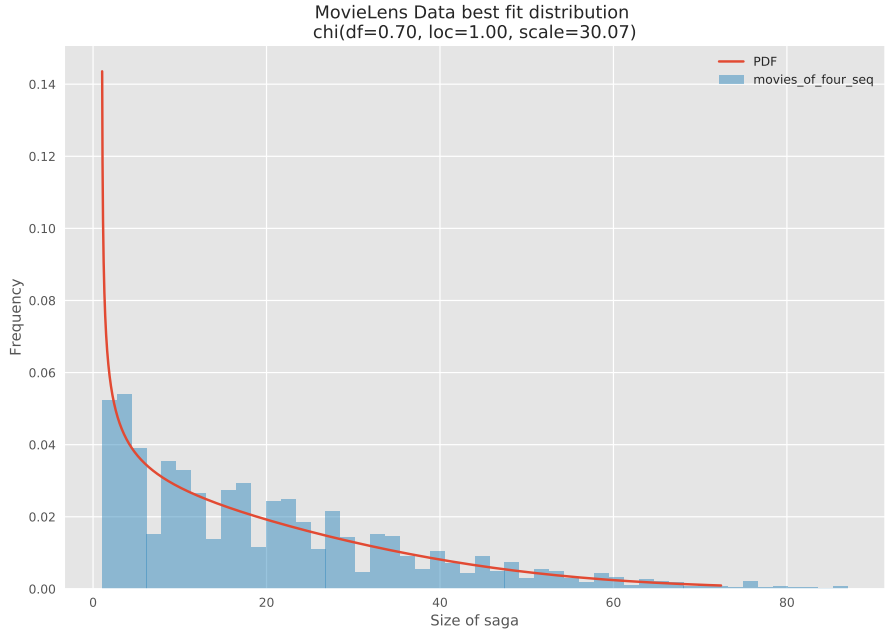

Figura 4.11: Número de subsecuencias de tamaño 4 vistas por los usuarios para sagas de distinto tamaño

La Figura 4.11 indica la distribución para sagas de 4 películas y subsecuencias de 4 películas. Como puede apreciarse en la distribución de los datos, es muy similar a la de sagas de 3 películas a excepción de que los usuarios con mayor número de películas vistas en orden son los que han visto entre 3 y 6 sagas de 4 películas. La razón es la misma que la comentada anteriormente solo que aquí el rango es más amplio. Para esta distribución de los datos, el modelo probabilístico que mejor se adapta es una distribución chi cuadrado con 0.7 grados de libertad.

De modo que, como comentamos al principio de el capítulo, hemos podido generar, aplicando el algoritmo SW a nuestros conjuntos de datos, una serie de resultados que siguen una cierta distribución y que hemos podido ajustar a un determinado modelo probabilístico. Gracias a eso, ahora disponemos

de uno de los dos elementos necesarios para el contraste de hipótesis. Si tuviésemos un sistema de recomendación de películas y realizáramos alguna serie de cambios en la forma de recomendar películas a los usuarios, podríamos volver a generar, aplicando el procedimiento seguido en este proyecto, un nuevo conjunto de datos que tuviese un modelo probabilístico similar a los que tenemos, pudiendo así realizar un contraste de hipótesis que nos permita verificar o refutar si los cambios que hemos realizado en nuestro SR han mejorado o no la tasa de usuarios que ven las películas en orden.

5

Conclusión y futuras mejoras

En la actualidad, los sistemas de recomendación son una herramienta muy importante y la cual se intenta mejorar continuamente. En cualquier empresa que ofrezca sus productos a través de la web, mejorar la recomendación de estos productos es indispensable para aumentar sus beneficios. Concretamente en el sector de la recomendación de películas existen muchas herramientas que recomiendan a los usuarios qué películas ver en función de sus gustos o de los gustos de usuarios similares a ellos. Pero como hemos podido comprobar en los resultados obtenidos en este proyecto, para películas pertenecientes a una saga, la forma en la que los usuarios ven dichas películas, no suele ser en el orden en el que son estrenadas. En definitiva esto puede ser debido al tiempo que pasa entre el estreno de una película y su continuación o por que muchos usuarios desconocen si una película tiene continuación, pero si que es cierto que para sagas de películas muy largas pocos usuarios las han visto de manera ordenada. Es por esto que es importante estudiar los patrones de visualización de los usuarios, con el objetivo de poder extraer información útil que pueda ser empleada en mejorar los sistemas de recomendación de películas.

La idea principal de este proyecto era la de obtener esta información desde cero, recopilando la información, procesándola y aplicando métodos que permitan extraer unos resultados consistentes que nos permitan entender mejor la forma en la que los usuarios ven las películas. De esta manera he podido comprender mejor todo el proceso y cómo este se desarrolla, así como conocer y aplicar nuevos algoritmos que inicialmente no se habían desarrollado para este propósito y que aún así han dado unos resultados más que satisfactorios.

En los resultados finales hemos podido comprobar que las sagas de películas que más se han visto en orden son aquellas que están compuestas por 3 y 4 películas. De este modo podemos analizar específicamente estas películas y ver qué tiempo ha transcurrido entre el estreno de una película y la siguiente de su misma saga. También se puede hacer un pequeño estudio del dinero invertido en publicidad para las películas de dicha saga o el coste de producción que estas tienen.

Como trabajo futuro se podrían emplear los datos recopilados y su distribución para realizar una serie de contrastes de hipótesis que demuestren si los cambios aplicados en nuestro sistema de recomendación han mejorado el número de sagas que los usuarios ven en orden. Estos sistemas de

recomendación se podrían mejorar analizando las sagas de películas que la mayoría de usuarios han visto en orden y viendo qué tienen en común dichas sagas, o probar a recomendar las sagas en un determinado orden en lugar de por puntuación o gustos. De esta manera, una vez se hayan aplicado los cambios, puede volverse a ejecutar sobre el nuevo dataset el algoritmo SW y ajustar los datos a un cierto modelo matemático. Si los nuevos resultados obtenidos se ajustan al modelo matemático de los resultados previos, se podrá realizar un contraste de hipótesis entre los resultados anteriores y los nuevos, pudiendo así ver si los cambios aplicados en el SR han dado resultados positivos a la hora de mejorar la tasa de usuarios que ve películas en orden.

También se podría aplicar directamente este algoritmo al proceso de recomendación de un usuario, comparando las secuencias de películas que este ha visto en una plataforma de visualización con la secuencia de películas de la saga a la que pertenecen las películas que ha visto y en caso de que dicha subsecuencia común no sea la secuencia completa, recomendar a dicho usuario las películas de dicha saga que no haya visto en el orden canónico que establece la plataforma de visualización.

Realmente en este proyecto no sabíamos que tipo de resultados se iban a obtener, ya que el algoritmo en el que se basa no estaba pensado para analizar este tipo de datos, por lo que podía darse el caso de que los resultados no aportaran información útil que pudiéramos utilizar para mejorar o implementar en los SR. Pero si es cierto que al final se obtuvieron unos resultados muy satisfactorios, con lo que podríamos aplicarlo a conjuntos mucho mayores de usuarios con un mayor volumen de información e investigar de manera más detallada los resultados que se obtuvieran.

Bibliografía

[1] "World Internet Users and 2019 Population Stats." https://www.internetworldstats.com/stats.htm. Accesed on 15 September 2019.

[2] B. Hallinan and T. Striphas, "Recommended for you: The netflix prize and the production of algorithmic culture," *New Media & Society*, vol. 18, no. 1, pp. 117–137, 2016.

[3] M. A. Ghazanfar, *Robust, scalable, and practical algorithms for recommender systems*. PhD thesis, University of Southampton, UK, 2012.

[4] S. Chakrabarti, "Focused web crawling," in *Encyclopedia of Database Systems, Second Edition*, 2018.

Acrónimos

SR Sistemas de recomendación.

SW Smith-Waterman.

SO Sistemas operativos.

kNN K-neares neighbours.

URL Uniform Resource Locator.

LCS Longest common subsequence.

HTTP Hypertext Transfer Protocol.


UAM
UNIVERSIDAD AUTONOMA
DE MADRID